Tampere University

Oskari Mikkilä

OPTIMAL HEDGING WITH CONTINUOUS ACTION REINFORCEMENT LEARNING

Master of Science Thesis
Faculty of Engineering and Natural Sciences
Examiner: Prof. Juho Kanniainen
April 2020

ABSTRACT

Oskari Mikkilä: Optimal Hedging with Continuous Action Reinforcement Learning
Master of Science Thesis
Tampere University
Industrial Engineering and Management
April 2020

This thesis proposes an application of state-of-the-art continuous action reinforcement learning for hedging European options in a market with discrete time rebalances and transaction costs. The hedging problem is defined as a utility function maximization problem. The utility function is defined as the expected change in the wealth minus a risk aversion parameter times the variance of the wealth. The utility function turns the hedging problem into a trade-off between expected wealth and variance. The proposed method is an application of the Twin Delayed Deep Deterministic Policy Gradients.

The method is tested in three different settings. Two of the settings are simulations of the asset and option prices. In the first setting the stock price process is a geometric Brownian process with constant volatility. The second setting uses the stochastic volatility model of Heston. SPX index prices and prices of call options written on SPX index from years 2012 and 2013 are used in the empirical data section. In the empirical setting this thesis proposes Sim-to-Real transfer learning: training the agent on asset price paths simulated by the Heston model and testing on historical data.

The proposed method performs better than selected analytical benchmarks in the stochastic volatility setting. In the constant volatility setting the reinforcement learning method has equal performance with the benchmarks. In the empirical testing section the method has equal performance with the benchmarks over the two-year period, performing better in more than 50% of the weeks. In general, this thesis shows that continuous action reinforcement learning can be used to balance the trade-off between cost and risk when hedging a European option.

Keywords: Options, Derivatives, Hedging, Reinforcement learning, Machine learning

TIIVISTELMÄ

Oskari Mikkilä: Optioiden optimaalinen riskienhallinta vahvistusoppimisella
Diplomityö
Tampereen yliopisto
Tuotantotalouden tutkinto-ohjelma
Huhtikuu 2020

Tässä diplomityössä esitetään tapa käyttää jatkuvan toimintoavaruuden vahvistusoppimista eurooppalaisten optioiden riskienhallintaan markkinoilla, joilla transaktioita voidaan suorittaa vain diskreetein aikavälein ja kaupankäynnistä veloitetaan transaktiokustannukset. Optioiden riskienhallinta käsitellään koetun hyödyn maksimointiongelmana. Hyötyfunktio määritellään seuraavasti: varallisuuden muutoksen odotusarvosta vähennetään riskiaversioparametrilla kerrottu varallisuuden varianssi. Tämä hyötyfunktio tarkoittaa kompromissia transaktiokustannusten ja avoimen riskin välillä. Esitetty malli käyttää kaksoisviivästyneitä syvädeterministisiä politiikkagradientteja (engl. Twin Delayed Deep Deterministic Policy Gradients).

Esitettyä mallia testaan kolmessa testiympäristössä, joista kaksi on simuloituja. Ensimmäisessä ympäristössä osakkeen hinta noudattaa geometrista Brownin liikettä ja kohde-etuuden volatiliteetti on vakio. Toisessa simulaatioympäristössä käytössä on Hestonin stokastisen volatiliteetin malli. Kolmannessa ympäristössä empiirisenä datana käytetään vuosien 2012 ja 2013 hintadataa SPX-indeksistä ja siihen kirjoitetuista optioista. Empiirisen datan kohdalla ehdotetaan siirto-oppimista: tekoälyagenttia koulutetaan Hestonin mallin avulla simuloiduilla hintapoluilla, mutta testaus suoritetaan historiallisella datalla.

Työssä esitetty tapa suoriutuu valittuja analyyttisia verrokkimetodeita paremmin stokastisen volatiliteetin simulaatiossa. Vakiovolatiliteetin simulaatiossa tekoälyagentti saavuttaa verrokkien kanssa yhtä hyvän tuloksen. Empiirisen datan kohdalla tekoälyagentti suoriutuu tasaisesti verrokkien kanssa kahden vuoden ajanjaksolla. Tekoälyagentti onnistuu hyötyfunktion maksimoinnissa vertailukohteita paremmin useammin kuin joka toisella viikolla. Yleisesti tulokset näyttävät, että jatkuvan toimintoavaruuden vahvistusoppimista voidaan käyttää eurooppalaisten optioiden riskienhallinnassa optimoimaan kompromissia kustannusten ja riskien välillä.

Avainsanat: Optiot, Johdannaiset, Riskienhallinta, Vahvistusoppiminen, Koneoppiminen

PREFACE

This has certainly been a challenging journey. I have been able to deepen my knowledge in two very interesting areas: hedging and state-of-the-art reinforcement learning.

A big thanks for the guidance to Prof. Juho Kanniainen.

Helsinki, 28th April 2020

Oskari Mikkilä

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

BS	Black-Scholes (model)
DDPG	Deep Deterministic Policy Gradients
GBM	Geometric Brownian motion
MDP	Markov Decision Process
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
TD3	Twin Delayed Deep Deterministic Policy Gradients
WW	Whalley-Wilmott method

1 INTRODUCTION

For banks and other financial institutions participating in the derivatives markets, hedging of derivative exposure is a critical activity. Hedging has an obvious role in the risk management of current exposures, where hedging has the purpose of limiting the risk to an acceptable level. Hedging also has importance from a pricing point of view. For a trading desk executing orders on behalf of its clients, the price charged or paid for a derivative is the expected cost of hedging plus a margin. This is pricing by replication.

Hedging and replication of contingent claims has been one of the most studied problems in the finance literature during the past 50 years. The most famous of the hedging and pricing methods is the dynamic replication by Black and Scholes (1973) and Merton (1973). Their replication method requires the most complete market of them all: the replicating position must be rebalanced in continuous time in absence of transaction costs.

In practice, the assumptions of the Black-Scholes model do not hold. The practitioners cannot adjust their hedge positions in continuous time, but rather have to choose a discrete time interval. Another problem arises from transaction costs. In a market with transaction costs, when the trading frequency increases, the sum of transaction costs rises.

This becomes a problem where the trader has to make a series of decisions between reducing risk and controlling costs. How often should the hedge position be adjusted? Should an adjustment be done now, or should you wait, bear the risk and save on transaction costs? This series of decisions is a problem of optimal control. This has traditionally been solved by utility functions. Utility functions, pioneered by Hodges and Neuberger (1989), give weights to both transaction costs and risk. The weights are determined by the hedger's risk appetite.

The principle of reinforcement learning is that an agent takes actions to maximize its expected rewards. The rewards can be engineered for a specific problem. When we set the agent's reward function to equal the hedging utility, the agent tries to solve the problem of finding the optimal action to maximize the expected hedging utility. The action in this case is if the agent should perform a transaction in the underlying asset, and if so, what kind of transaction. In this thesis, reinforcement learning is used to provide a machine learning based hedging method for European options, with the aim of optimizing the hedging actions in a discrete time market with transaction costs.

Reinforcement learning is a sub field of machine learning that has developed rapidly in

the last decade. Lately, reinforcement learning has achieved some notable milestones. Beating the best human players in a game of Go was for long a challenge too tough for neural networks, but recently the first AI to do so was based on reinforcement learning methods (Silver, Lever, Hees et al. 2016). In the realm of video games, a team of five neural networks was able to achieve human level performances in Dota 2 by training against itself hundreds of millions of times (OpenAI 2018). Outside of games, reinforcement learning is known, among other things, for the applications in robotics (Polydoros and Nalpantidis 2017).

Applications of supervised learning in finance, and particularly option pricing and hedging, have gotten a lot of attention since the early 1990s (Ruf and Wang 2019). Reinforcement learning on the other hand has been with relatively little attention (Kolm and Ritter 2019). There are some areas in finance with a handful of studies, like portfolio allocation (e.g. Jiang, D. Xu and Liang 2017; Yu et al. 2019).

Previous work of reinforcement learning approaches to hedging and pricing of options include methods similar to what is proposed in this thesis. Halperin (2017) uses Q-learning of Watkins' (1989) in a discrete time Black-Scholes world, but they do not consider transaction costs. Buehler et al. consider a portfolio of derivatives with an risk measure approach (2018; 2019). Ritter and Kolm (2018) introduce a discrete action method based on the geometric Brownian stock price process, where transaction costs are quadratic. J. Cao et al. (2019) extend this and also consider a setting with stochastic volatility.

1.1 Objectives of the Thesis

The topic of this thesis is to use continuous action reinforcement learning for option hedging. The topic is relevant and timely for multiple reasons. The interest among hedging practitioners for smarter, automated solutions is as strong as ever. There number of studies applying reinforcement learning to hedging options is low, and these studies focus on discrete action reinforcement learning. In theory, converged continuous action methods hold a natural advantage over discrete action methods, as they allow better hedging precision. Lastly, as far as we are aware, none of the previous studies show hedging performance on empirical data.

The objective of this thesis is to develop a continuous action reinforcement learning application for hedging, where hedging is expressed as a utility function maximization problem. The proposed method is only one of the possible solutions, as there are numerous other methods for continuous action reinforcement learning and the hedging problem can be expressed in multiple ways. This thesis does not intend to compare between other possible solutions.

There are two research questions for this thesis:

RQ1: How does the proposed continuous action reinforcement learning hedging method perform against analytical methods on simulated asset price paths?

RQ2: How does the proposed reinforcement learning agent perform when taught on on simulated data and tested on real data?

The purpose of the research questions set is to assess the performance of the proposed method: can a continuous action reinforcement learning agent learn hedging policies that maximizes the utility function. The research questions underline the importance of this thesis. As far as we are aware, there are no studies that assess the performance of continuous action reinforcement learning for hedging utility problem. Earlier studies use either discrete actions (e.g. J. Cao et al. 2019; Halperin 2017; Ritter and Kolm 2018) or formulate the hedging problem in a different way (Buehler, Gonon, Teichmann and Wood 2018). This thesis aims for an in-detail description of a working solution of applying reinforcement learning for hedging. Studies on reinforcement learning have suffered from reproducibility problems (Islam et al. 2017). To enable any reproduction attempts, this thesis aims to document every essential implementation detail.

1.2 Organization of the Thesis

This thesis is structured as follows. Chapter 2 gives a brief introduction to reinforcement learning, and goes more in depth into some relevant reinforcement learning methods. Chapter 3 introduces the relevant financial models, the Black-Scholes and the Heston model, and discusses earlier work done on optimal hedging in a market with frictions, both analytical and reinforcement learning approaches. Chapter 4 goes in depth on the setting and continuous action approach to optimal hedging. Chapter 5 shows how the method performs in three different settings and chapter 6 concludes the thesis.

2 REINFORCEMENT LEARNING

The field of machine learning is often divided into just two sub fields, supervised and unsupervised learning. These two fields continue to receive the most attention in the machine learning literature. Supervised learning is a task of finding a function F which maps input data vector x to an output vector y. Supervised learning methods require that both the input vector and output vector are provided and only the function F is unknown. If the function mapping is a success, the function can then be used to generate outputs for unseen input data. Examples of use cases for supervised learning methods are text and image classification and regression analyses.

As opposed to supervised learning, unsupervised learning does not have any target that the input is mapped to, but the goal is instead to learn some features of the data. Unsupervised learning methods can be used for clustering problems, where data is grouped into a number of clusters, without it being necessary to understand what the data is. Clustering can be used for example to form customer segment groups or to build recommender systems. Unsupervised learning can also be used to spot outliers in a data set, which is known as anomaly detection.

Third sub field of machine learning is reinforcement learning, which lies somewhere in-between supervised learning and unsupervised learning. The goal in reinforcement learning is to create algorithms that can learn to take the optimal action in an environment. The optimal action is defined as an action that maximizes the expected lifetime reward (Silver, Lever, Heess et al. 2014). The difference between supervised learning and reinforcement learning is that a supervised learning solution would try to teach an agent which action to take given the current state, and a reinforcement learning solution would tell the agent to try out different strategies and see which is the best. (Sutton and Barto 2018)

Reinforcement learning systems have two entities which interact with each other; an agent and an environment. Agent is the entity that takes actions and tries to learn a *policy* π. Policy is a set of rules or weights which defines which action to take given the situation the agent is in. After an action is taken, the environment reacts to the action. The state of the environment changes based on the action taken and sends two kinds of information back to the agent. The environment returns an observation of the updated state, and calculates and returns a reward which quantifies the quality of the action taken. After receiving a reward and a state observation, the agent will continue and take next action. Figure 2.1 displays the cyclical interaction between the agent and the environment.

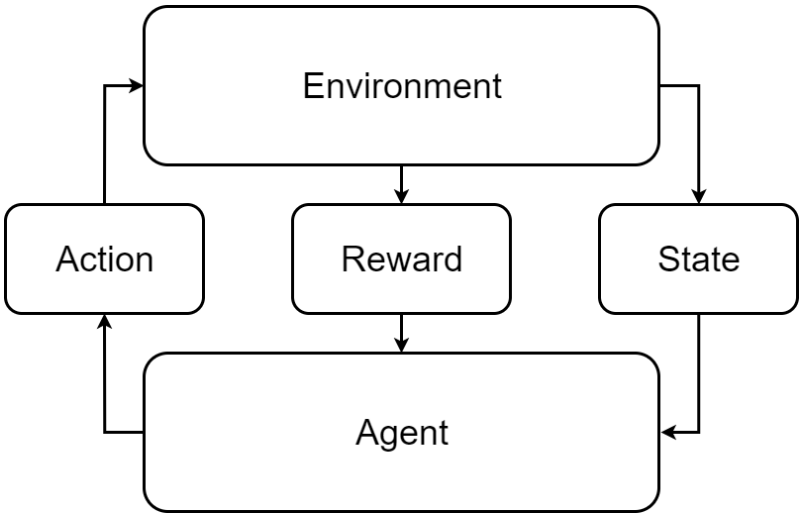

Figure 2.1. *Reinforcement learning system, where an agent interacts with an environment*

This continues until a terminal state is reached. The period between the start and terminal state is called an episode. An episode can end for different reasons depending on the problem being learned. In a video game, an episode can end and the terminal state is reached if the player dies or if the time runs out. For a stock trading algorithm, the episode could end if the algorithm loses all available cash. Some problems do not have a natural ending for an episode and instead an arbitrary limit for the maximum number of time steps can be set.

There are two types of actions in reinforcement learning problems. Discrete actions have a known, limited space of available actions. Continuous actions might have upper and lower limits, but any action in between can be chosen. For example, a discrete action could be if an agent should turn the car left or right, while a continuous action tells how much to steer the wheel.

The goal in most of the reinforcement learning problems is to learn a policy that maximizes the received reward over the entire episode. This gives an additional dimension to the problem of choosing the correct action: the optimal action in the current state might result in a negative reward immediately but lead to large positive rewards later. On the other hand, an action might lead to a large immediate reward but be disastrous on the long run.

The balance between immediate and future rewards is achieved with a familiar method from the world of finance: by discounting the rewards. The further away in the future the reward is, the lower its present value. The sum of discounted future rewards G_t is

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-1} R_T \tag{2.1}$$

or

$$G_t = \sum_{t=0}^{T-1} R_{t+1} \gamma^t \tag{2.2}$$

Where r_t is the immediate reward received at time t and γ is the discount rate. The discount rate is problem specific and should be tuned to find the best results.

The idea of using rewards to quantify the quality of actions is intuitive and often compared to how human or animal minds work. After taking an action and receiving a reward, it is intuitive to increase the probability of taking this action in similar situations. Vice versa, actions for which you will get penalized for will be taken more sparsely. (Ciaburro 2018) The agent has an inner policy which guides it to evaluate which action should be taken. How to update the policy based on actions taken and rewards received depends on the chosen reinforcement learning method, of which there are multiple. Designing the reward system is often one of the biggest challenges in itself.

Agents must explore the possible action space to learn which actions are good and which not. There is a trade-off between choosing the action that has worked in the past and exploring the action space in hopes of learning something even better. This problem is known as the exploration-exploitation dilemma. Without any exploration policies can easily fall into local value maximums. However, if you are only exploring random actions the agent will most likely not be able to maximize the lifetime reward. For discrete action spaces this dilemma is often solved by the epsilon greedy strategy: choose a random action with probability ϵ, and the best, *greedy*, action with probability $1 - \epsilon$. By changing the value of the hyper parameter ϵ and decaying it over time, a semi-optimal amount of exploration and exploitation can be found. For continuous action spaces exploration can be done by adding a random number from for example the Gaussian distribution.

Cornerstones of reinforcement learning algorithms are state value and action value functions. Most of the algorithms use one or both. State-value function can be thought to be the value of being in state *s*. The value of being in a state equals to the expected sum of discounted future rewards the agent will earn, if it starts from the state *s* and follows a policy π afterwards.

$$V(s_t) = E_\pi(G_t|s_t) \tag{2.3}$$

The calculation of the expectation differs between reinforcement learning methods. Similarly, we can define the value of being in state s_t and taking an action a_t and following a policy π afterwards as

$$Q(s_t, a) = E_\pi(G_t|s_t, a) \tag{2.4}$$

this is the action-value function. These functions and their values can be used to compare actions and states, and thus learning the functions is a central problem in RL tasks.

A state observation which contains all relevant information to find the optimal action is a one which fulfills the Markov property. The concept of the Markov property is essential in reinforcement learning, as the action is always selected based on only the information in the state observation. (Sutton and Barto 2018) If the state observation does not contain enough information, most likely the agent will fail to learn an optimal policy. A reinforcement learning problem which holds the Markov property is a Markov decision process ("MDP").

2.1 Simple RL methods for discrete action spaces

The simplest of all reinforcement learning methods are the tabular methods. These methods form a table where the learned value for each state and action pair can be stored. The methods are only usable when the problem is very simple: the table will grow larger and larger when the number of possible states and actions increase.

When forming an action-value table the value for each s, a pair can be calculated as the mean reward from gathered experience. For a given state s_t and all possible actions, the Q-value for an action a_t is

$$Q(s_t, a_t) = \frac{R_{e1} + R_{e2} + ...R_{eN}}{N} \tag{2.5}$$

Where R_{ei} is the total reward for episode *i* as in equation (2.2). The Q-function then satisfies equation (2.4): it is the expected reward for taking an action a_t in state s_t.

Monte Carlo methods use similar s, a pair reward averaging with the help of learning rate α

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(G_t - Q(s_t, a_t)\right) \tag{2.6}$$

Where we move from previous Q-function estimate for the future reward to the direction of the experienced reward. Monte Carlo methods can solve simple problems where the action or state space dimension is low enough.

2.2 Temporal Difference

The logic behind Temporal Difference methods is one of the central developments to enable the progress of RL. Temporal Difference was developed by one of the reinforcement learning pioneers, Richard Sutton (2018). Temporal Difference is based on the idea that the update of state or action value functions use bootstrapping: the updates are based on estimates of future rewards. State value can be expressed with the reward received now and with estimate of next state as

$$V(s_t) = R_{t+1} + \gamma V(s_{t+1}) \tag{2.7}$$

(Sutton and Barto 2018). As the value of V converges the estimation of the next state's value gets closer to the real value.

When we insert above to a similar equation as Eq. (2.6), we get an update rule for the state value function

$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right] \tag{2.8}$$

(Sutton and Barto 2018). Bootstrapping gives an efficiency boost, as the value function can be updated before the episode finishes.

Probably the most well known reinforcement learning algorithms is the Q-learning algorithm by Watkins (1989). Q-learning defines an action value function

$$Q\left(s_t, a_t\right) = R_t + \gamma \text{max}\, Q\left(s_{t+1}, a_{t+1}\right) \tag{2.9}$$

which is the immediate reward received plus the discounted value of all future rewards obtained by following the optimal policy.

The update rule for the Q-function follows the temporal difference update rule from Eq. (2.8)

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha \left[R_{t+1} + \gamma \underset{a}{\text{max}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{2.10}$$

Taking the maximum Q-value over the actions in the next state mean that the optimal policy is followed after current action. This allows Q-value calculation for any action.

Multiple improvements to Q-learning have been presented. Strehl et al. (2006) introduced Delayed Q-learning, where the Q-function for each (s, a) pair is updated only every n visits. One reason for a possible poor performance of Q-learning is that the action values can be heavily over estimated. To combat this Hasselt (2010) introduced double Q-learning, a method that utilizes two Q-functions, Q^A and Q^B. Q^A is used for selecting the action. When updating Q^A, instead of using $Q^A(s_{t+1}, a_{t+1})$, the value from Q^B is used. The second function works as an unbiased estimator and reduces overestimation. Plugging in a neural network for the Q-function approximation in plain Q-learning leads to unstable learning. This happens because the sequential observations are correlated. In the Deep Q-learning of Mnih et al. (2013), this was solved by storing the observations in memory and updating the Q-function by sampling random observations.

2.3 Actor Critic

Actor-critic methods separate the policy and the evaluation of the agent's actions into two entities. The *actor* learns a policy, and the *critic* is the value function and the one evaluating the policy. (Sutton and Barto 2018) The actor-critic system is illustrated in figure 2.2. Actor-critic methods are based on an idea that critic gives feedback to the actor, which updates its policy based on the feedback given by the critic. The critic updates its own estimates based on the estimation error. Variations on how the updates are performed depend on the type of the actor-critic algorithm. Actor-critic methods utilize temporal difference learning in the critic.

Earlier actor-critic methods were on-policy algorithms (Sutton and Barto 2018), but newer algorithms like Silver et al.'s DPG (2014) have made off-policy learning possible. On-policy algorithm is an algorithm where the policy updates and value estimates depend on the current policy. Off-policy does not have a dependency on the current policy, but instead often uses the optimal policy for learning. Off-policy makes it possible to use *replay buffers* to store experience for later use. Replay buffers acts as a memory, where state, action, reward and next state tuples are saved. These memories can then be drawn

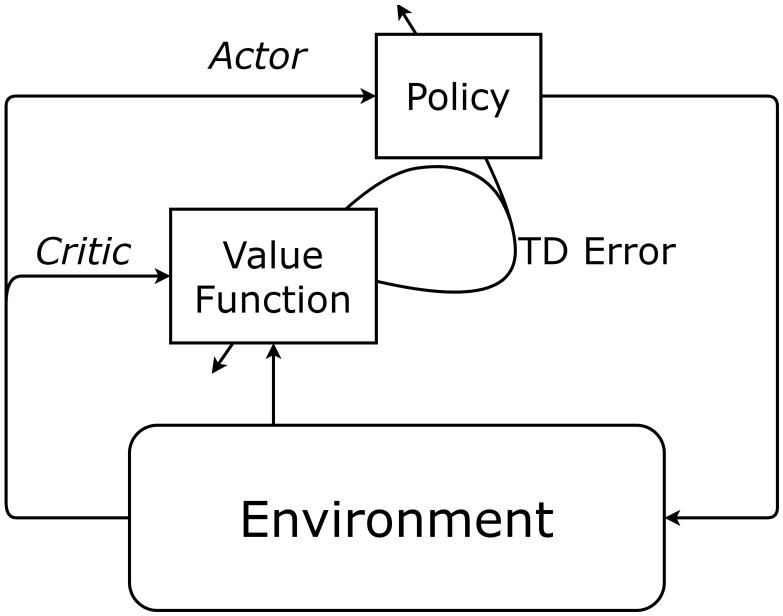

Figure 2.2. *Actor-critic system, modified from (Sutton and Barto 2018)*

from the replay buffer over and over again to improve sample efficiency.

2.4 Deterministic Policy Gradients

Traditional solution for continuous action spaces has been to learn a policy represented by a probability distribution, a *stochastic policy*

$$\pi_\theta(a_t|s_t) = \mathbb{P}[a_t|s_t, \theta] \tag{2.11}$$

which most often is parameterized by a mean and a variance, from which an action can be drawn. Silver et al. introduced an algorithm for *deterministic policies* (2014),

$$a_t = \mu_\theta(s_t) \tag{2.12}$$

Prior to their work it was believed that deterministic policy gradients can be only used when the model of the environment is known. Largest difference between the stochastic and deterministic gradients is that the stochastic gradient integrates over both action and state spaces, whereas deterministic gradient integrates over state space only, which makes deterministic gradient more sample efficient. However exploration over the action space is not guaranteed in deterministic policies, so exploration noise should be added.

Policy gradients are based on the idea that the policy parameters θ are adjusted by calculating the performance gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\ \pi_0}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)] \tag{2.13}$$

Multiple algorithms have been introduced to estimate the action-value function $Q^\pi(s,a)$. (Silver, Lever, Heess et al. 2014)

Perhaps the best known deterministic policy gradient algorithm is the Deep Deterministic Policy Gradients ("DDPG") by Lillicrap et al. (2015). DDPG introduces similar improvements to deterministic policy gradients as Deep Q-learning did to Q-learning, and DDPG is known as the equivalent of Deep Q-Learning for continuous action spaces.

DDPG is an off-policy actor-critic algorithm. DDPG learns a Q-function, which is used as the critic. DDPG updates the critic by minimizing the loss between the current Q-value estimates and target values

$$L = \frac{1}{N}\Sigma_i \left(y_i - Q(s_t, a_t|\theta^Q)\right)^2 \tag{2.14}$$

where N is the batch size drawn from the replay buffer. The policy is updated according to the policy gradient theorem in Eq. (2.13).

To combat drastic changes in the policy and action-value networks, DDPG uses target networks with polyak averaging

$$\theta_{\text{target}} \leftarrow \tau\theta_{\text{target}} + (1-\tau)\theta \tag{2.15}$$

where the network weights are slowly copied into the target networks and the rate of copy is controlled by the parameter τ.

2.5 Twin Delayed Deep Deterministic Policy Gradients

Even if DDPG gained a lot of attention and popularity for continuous action problems, it performs poorly on some environments and has some issues. The algorithm was found to be unstable in some environments and hyper parameter sensitive, like so many other reinforcement learning algorithms. Most of the problems found in DDPG were found to arise from the over estimation of Q-values in the critic network. This leads to the policy exploiting the over estimations. Similar issues have been seen in Q-learning methods. (Fujimoto, Hoof and Meger 2018)

To overcome these issues an extended version of DDPG was introduced in 2018 by Fujimoto et al. (2018). The algorithm is named Twin Delayed Deep Deterministic Policy Gradients ("TD3") after the extensions it brought to DDPG. TD3 combines some of the greatest achievements and tricks introduced in other deep reinforcement learning methods, such as using two independent critic networks. Idea for double critic comes from Double Q-learning. The overestimation of Q-values is battled by taking the smaller Q-value from the two critic networks for the target value

$$y_i = R_t + \gamma \min_{i=1,2} Q_{\theta_{\text{target},i}}(s_{t+1}, \pi(s_{t+1})) \tag{2.16}$$

this target value is used in Eq. (2.14). The term $\pi(s_{t+1})$ is the target action.

Another common problem with deterministic policies occurs when the Q-values have large peaks in some state and action areas. The policy quickly tries to get to that peak

more often. In TD3 target policy smoothing is done to smooth out these peaks. The policy smoothing is done by simply adding noise to the target action. The target action becomes

$$a_{\text{target}} = \text{clip}\left(\pi(s_{t+1}) + \epsilon, a_{\text{low}}, a_{\text{high}}\right) \tag{2.17}$$

which replaces the target action in Eq. (2.16). The policy smoothing resembles the update rule of SARSA (Sutton and Barto 2018). Last improvement introduced in TD3 is familiar from delayed Q-learning. Actor network is updated less frequently than the critic networks to prevent the actor from exploiting the critic estimates. The original authors update the actor for every second critic update.

3 OPTION PRICING AND HEDGING

In this chapter relevant option pricing models, Black-Scholes and Heston models, will be introduced. We will also review the hedging methods of these models and we will introduce few of the hedging methods that have been developed to overcome the shortcomings of these hedging methods, namely ones that have been developed to deal with transaction costs. Last, we will review previous machine learning and reinforcement learning methods that have tried to solve the question of optimal hedging.

3.1 Black-Scholes

The Black-Scholes ("BS") model, published in 1973 by Black and Scholes (1973), is the most well-known option pricing model. Although other models have arguably surpassed BS model in the practitioners' eyes, it is still widely used, taught and studied. The model holds some attractive properties: the pricing formula for European options is relatively simple and in closed form, and there is only one parameter that cannot be observed in the market, the volatility of the underlying asset.

The Black-Scholes model assumes certain conditions to exist in the market:

1. Hedge balancing can be done continuously
2. The risk free rate is known and constant
3. The stock price process follows Geometric Brownian motion
4. The volatility of the underlying asset is constant
5. The underlying asset pays no dividends
6. The markets are frictionless, there are no transaction costs
7. Unlimited short selling is allowed

Even though the model became popular among practitioners in the industry, the assumptions have been criticized for being idealistic and non-representative of the market. In the four decades after it was first published, a lot of work has been done to relax some of the assumptions. Extensions of the model where the dividend assumption can be relaxed by using dividend yield have been introduced by for example Rubinstein (1976).

Geometric Brownian motion for stock price process can be expressed with the stock price

drift μ, volatility σ and a Wiener process dW_t as

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{3.1}$$

The key idea behind the BS model is that given the aforementioned constraints, an option can be perfectly replicated by continuously balancing a portfolio of holdings in the underlying asset and the risk free asset. To replicate an option perfectly, holdings in the underlying asset must equal the partial differentiation of the option price in respect to the underlying asset price. The value of this portfolio is

$$\Pi = -f + \frac{\partial f}{\partial S} S \tag{3.2}$$

where f is the option price and S is the underlying asset price. The change of value in time interval Δt is

$$\Delta\Pi = -\Delta f + \frac{\partial f}{\partial S}\Delta S \tag{3.3}$$

By applying Ito's lemma we arrive at

$$\Delta\Pi = \left(-\frac{\partial f}{\partial t} - \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2\right)\Delta t \tag{3.4}$$

The change in value is risk free, and by no-arbitrage constraint this change must equal the risk free rate. That leads to

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf \tag{3.5}$$

which is the Black-Scholes differential equation. Full derivation can be obtained from Hull (2012). Above differential equation can be solved for European call options by substituting $f = \max(S - K)$ at expiration. Call option price can then be obtained from equation

$$C = S\mathcal{N}(d_1) - Ke^{-rt}\mathcal{N}(d_2) \tag{3.6}$$

where S is the current stock price, K the strike price, r the risk-free rate, t the time to expiry and $\mathcal{N}(.)$ is the standard cumulative probability distribution taken from

$$d_1 = \frac{\ln\frac{S}{K} + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} \tag{3.7}$$

and

$$d_2 = \frac{\ln\frac{S}{K} + (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T} \tag{3.8}$$

Hedging an option by replicating the option's price moves by holding some amount of the underlying asset is called *delta hedging*. Delta (Δ) is the partial derivative of an option's price with respect to the underlying asset's price. The BS model is built on the dynamic

replication principle. The BS delta is given by

$$\Delta = \frac{\partial f}{\partial S} = \mathcal{N}(d_1) \tag{3.9}$$

Thus a perfect replication of an European call option in the Black-Scholes world can be achieved by holding $\mathcal{N}(d_1)$ stocks and rebalancing continuously. Continuous rebalancing means that the hedger's position in the underlying asset must be updated in continuous time. The updates are required when the delta changes, i.e. when any of the parameters of d_1 change. The largest differences in the delta are caused by the asset price S moving. When the position is balanced continuously, the change in the value of the portfolio consisting of the option, the position in the underlying asset and in the risk free asset will always be zero. The changes in the option price are offset by the changes in the underlying asset's price and the risk free asset. As the change of the portfolio's value is zero, the variance of the value of the portfolio will be zero, and the portfolio is risk free.

In practice, continuous rebalancing is impossible, and the hedge must be rebalanced at discrete time intervals. This will introduce tracking error to the portfolio. When price of the underlying asset moves, the delta changes. In general, the longer the period between balances is, the higher the moves in the underlying asset's price are. This results in higher variance in the value of the portfolio when the balance intervals are longer. Thus, the practitioner must make a decision between having to rebalance often and bearing variance.

3.2 Heston model

Since the publication of the Black-Scholes model, it has been noted that the model is unable to capture the volatility seen in the quoted option prices in the market. In market quoted prices, the observed volatility is not a constant but rather a function of both the strike price and expiration. Most often the implied volatility increases when the option goes deep in the money or out of the money. This is called is called the volatility smile, which is illustrated in figure 3.1. When considering options with different expirations we see that the smile becomes a 3-D volatility surface.

Multiple models have been introduced to capture the complex volatility in the market prices. Perhaps the most well known is the model introduced in 1993 by Steven Heston (1993), which is able to capture most of the skew and smile patterns observed in the option market prices (Albrecher et al. 2007).

The stock price process of the Heston model is similar to the GBM in Eq. (3.1) used in the Black-Scholes model

$$dS_t = \mu S_t dt + \sqrt{v_t} dW_{1,t} \tag{3.10}$$

but the key difference is that the variance v_t is a stochastic process. The instantaneous

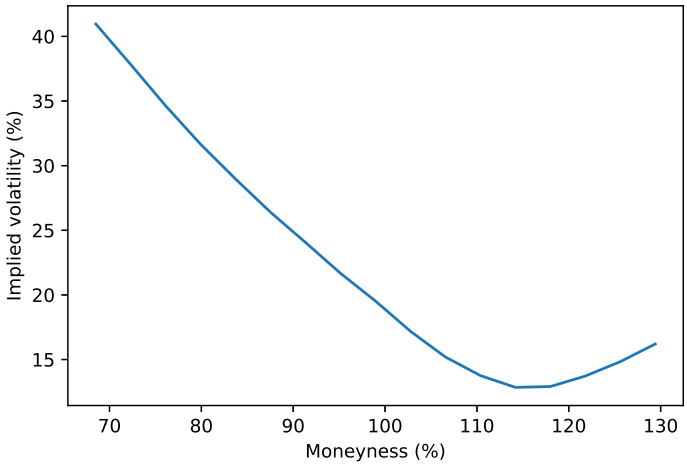

Figure 3.1. *SPX option volatility skew as seen on 25 January 2012.*

Table 3.1. *Heston model parameter descriptions*

Parameter	Description
θ	The long time average volatility
ρ	The correlation between the asset's price and volatility
κ	The rate at which the instantaneous volatility returns to the long time average
ϵ	The volatility of volatility
v_t	The instantaneous variance

variance is modelled as a CIR process (Albrecher et al. 2007)

$$dv_t = \kappa\left(\theta - v_t\right)dt + \varepsilon\sqrt{v_t}dW_{2,t} \tag{3.11}$$

which is a mean reverting process: the instantaneous variance reverts to the long time average θ with the rate of κ. In above two equations the two Wiener processes $W_{1,t}$ and $W_{2,t}$ have ρ correlation. The parameters of the two processes are summarized in table 3.1.

The Heston model parameters need to satisfy a condition

$$2\kappa\theta > \epsilon^2 \tag{3.12}$$

known as the Feller constraint to ensure that the variance process stays positive. (Albrecher et al. 2007)

Heston call price then gets form

$$\begin{aligned} C_t(K) &= e^{-rT}\mathbb{E}\left((S_t - K)^+\right) \\ &= e^{x_t}P_1(x, v, T) - e^{-rT}KP_2(x, v, T) \end{aligned} \tag{3.13}$$

where $x_t = lnS_T$ and where P_1 is the probability of the option expiring in-the-money conditional on the value of the stock, and P_2 represent the probability of expiring in-the-money conditional to the volatility. (Albrecher et al. 2007)

3.3 Hedging under transaction costs

As the Black-Scholes model assumes, if the hedge is rebalanced continuously, the transaction costs will eventually grow to infinity, no matter how small the transaction costs are (Leland 1985; Zakamouline 2002). Of course rebalancing in continuous time is impossible in practice, so rebalancing has to be done at some discrete intervals. Thus the simplest discrete time hedging strategy is to hedge according to the Black-Scholes delta, but only balance at discrete intervals. Choosing the interval is a trade off between risk and cost. The longer the interval, the lower the number of transactions and the lower total transaction costs, and vice versa (Leland 1985; Zakamouline 2006). Discrete hedging algorithm will then need to address a balance between transaction costs and hedging error.

Discrete time introduces another source of risk to the replication problem, even if the underlying asset's price process was known, the moves in price will always present some mismatch in the option's price and the replicating portfolio's price. This mismatch is called slippage or tracking error.

Hedging in a discrete time market with frictions has been tried to solve by introducing utility functions that are maximized or minimized. Pioneering of utility based hedging is often credited to Hodges and Neuberger (1989). The idea behind utility function methods is to find a price, which gives the option writer enough premium to offset the risk she takes. The writer is thus indifferent to writing the option or not as the expected utility is zero. (Zakamouline 2002) Utility based pricing has an obvious drawback. It cannot provide a unique arbitrage-free price like Black-Scholes can, as both the writer and holder of the option have different utility functions, and the utility functions are based on market participants' risk appetite.

One of the earliest work on optimizing transaction costs and hedging was written by Leland (1985), who introduces a fairly simple extension to the Black-Scholes model that can be used to hedge in discrete time. Leland's method uses modified volatility

$$\hat{\sigma} = \sigma \pm \sqrt{\frac{2}{\pi}} \frac{\kappa}{\sigma\sqrt{\Delta t}} \tag{3.14}$$

where σ is the original (implied) volatility, κ is the proportion of transaction costs of the underlying asset's face value, and Δt is the rebalancing frequency. Sign of the adjustment depends on the option type and hedger's position, plus for short call or long put position, minus otherwise.

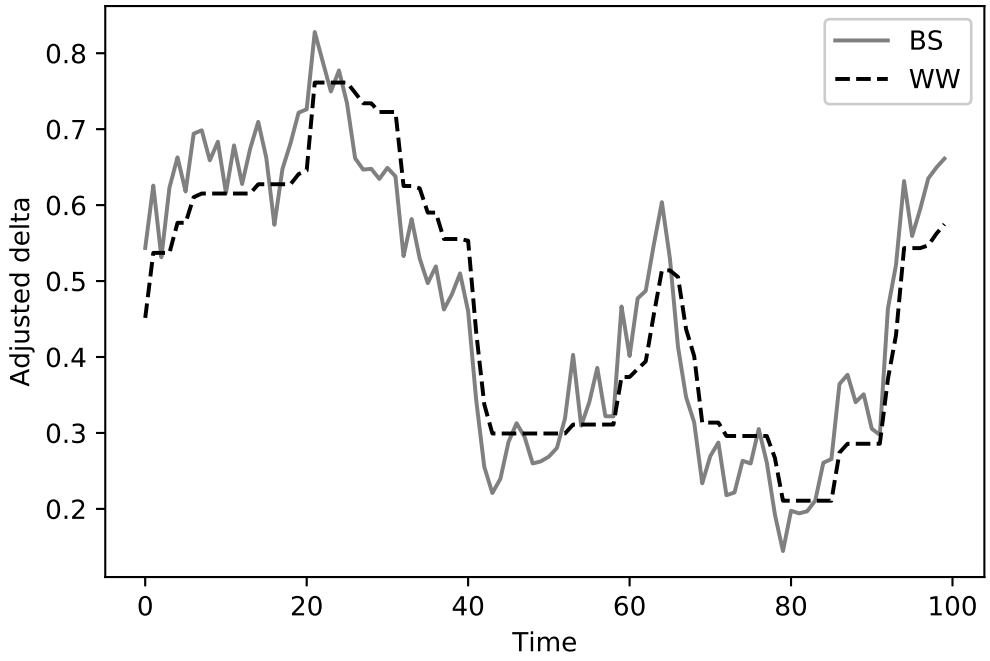

Figure 3.2. *An example of a Whalley Wilmott ("WW") hedge compared to a Black-Scholes delta hedge ("BS"), when the risk aversion parameter λ is 0.5 and transaction costs κ are 0.2 %.*

Leland's modified hedge ratio can be then calculated with help of Eq. (3.9)

$$\hat{\Delta} = \mathcal{N}\left(d_1\left(\hat{\sigma}\right)\right) \tag{3.15}$$

κ and Δt need to be small and the ratio $\kappa/\sqrt{\Delta t}$ is of order one. (Barles and Soner 1998; Leland 1985)

Alternative solutions to optimal utility based hedging are so called *delta tolerance* strategies. These strategies are based on an idea that the hedge should only be updated if the current hedge ratio moves outside a specified delta range. This delta range is called the *no transaction zone*, as no transactions are done while the current hedge ratio is inside this zone (Zakamouline 2006).

Whalley and Wilmott (1997) introduced a solution for delta tolerance strategy in their 1997 paper. The distance allowed for the current hedge is displayed by a range where the Black-Schole delta is in the center. The distance of the bounds from the center depend on the speed of change in delta when the underlying asset price moves, the *gamma*. Gamma is a partial derivative of delta in respect to the underlying asset's price,

$$\Gamma = \frac{\partial^2 f}{\partial S^2} \tag{3.16}$$

i.e. a second derivative of the derivative's price in respect to the asset price. In the

Black-Scholes world this becomes

$$\Gamma = \frac{\mathcal{N}'(d_1)}{S\sigma\sqrt{T}} \tag{3.17}$$

(Hull 2012) where d_1 comes from Eq. (3.7) and $\mathcal{N}'$ is the probability density function for a standard normal distribution

$$\mathcal{N}'(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{3.18}$$

Taking the BS-delta from Eq. (3.9) and adding (subtracting) the boundary distance term we get the delta range

$$\Delta_{\pm} = \frac{\partial f}{\partial S} \pm \left(\frac{3}{2} \frac{e^{-r(T-t)} \kappa S_t \Gamma_t^2}{\lambda} \right)^{\frac{1}{3}} \tag{3.19}$$

where κ is again the proportional transaction costs and λ is a risk aversion parameter (Whalley and Wilmott 1997; Zakamouline 2006).

Whalley-Wilmott method is intuitive, as can be seen from figure 3.2. A small change in delta does not trigger a rebalance. The Black-Scholes delta has a lot of variance and rebalancing the hedge to achieve delta neutral position would lead to a high sum of transaction costs. By allowing the portfolio some delta exposure the number of transactions is visibly lower. The inclusion of gamma ensures that the bounds are closer if the gamma exposure is high, which could lead to sudden increases in the portfolio's delta exposure. Alternative solution was given by Barles and Soner (1998) who propose using adjusted volatility.

3.4 Hedging under stochastic volatility

Common way of hedging among practitioners is to calculate the Black-Scholes implied volatility from the quoted market prices, and calculate the delta with the implied volatility. This method is referred to as *practicioner delta-hedging* by Hull and White (2017). This practice is not an optimal one to minimize the variance of the position. The variance is not minimized as there exists a negative correlation between the underlying asset price and its volatility. When the underlying asset's prices decreases, the volatility increases, and vice versa. (Hull and White 2017).

Renault and Touzi (1996) show that the Black-Scholes hedge leads to an under hedged position for in-the-money options and to an overhedged position for out-of-the-money options. For at-the-money options the Black-Scholes hedge gives perfect partial hedged position. These effects relate to the volatility smile.

Although the stochastic volatility pricing models have been accepted as more accurate than the constant volatility Black-Scholes model, the same does not apply to delta hedg-

ing. Taking a simple partial derivative of the price with respect to the underlying asset price has been shown to perform poorly. However, when the expected size of the move is taken into account, the performance improves. (Bakshi, C. Cao and Chen 1997; Hull and White 2017)

Alternative approach to forming a hedge with a position in the underlying asset is to hedge the volatility exposure with another option or derivative. One can calculate the positions volatility exposure and take an offsetting position. Hedging with derivatives rather than the underlying asset is often too costly to be considered (Ritter and Kolm 2018).

3.5 Previous reinforcement learning approaches to option hedging

Halperin (2017) introduce a method which they call QLBS, Q-Learning in Black-Scholes world. They use Q-Learning for pricing options using risk adjusted returns. QLBS is model free as the approach is not necessarily dependent on any specific stock price model or process, but only on samples drawn from those distributions. In their work they show an application on both simulated stock price process and how to apply it to real trading data. Halperin's method neglects transaction costs and assumes that the Black-Scholes assumptions hold except for continuous rebalancing. Halperin's method has the *optimal hedge* as an action. As the authors note, an interesting result from their method is that the optimal hedge is an argument to a function that gives the option price.

The standard version of Q-Learning requires calculating the action-value for all possible actions to find the correct one. This is not very suitable for problems where the action space dimensionality is high or continuous. To find the optimal hedge the action space has to be discretized. Black-Scholes delta takes values between 0.0 and 1.0, which can be used as limits for the action space. For example, by choosing actions with 0.01 steps from this range you would end up with 101 actions. Halperin states that they used discrete action space, but does not specify the action range used. They note that for discrete state space this function can be solved analytically as the stock price transition probabilities can be calculated. For a continuous state space Monte Carlo can be used.

Ritter and Kolm (2018) address hedging in a setting of discrete time rebalancing and transaction costs. They do not assume linearity of transaction costs but present that their method could be used with a transaction cost function of any kind. They present a method which they say could be used for any pricing or simulation method, similarly as Halperin's.

They teach the agent to trade an option with fixed maturity and strike price. The agent has a long position in an option which must be held until expiration. Hedging is done by trading the underlying asset. They have a continuous state space, while Halperin had discrete. They also discretize the action space to integer number of shares. State is represented with current asset price, time to expiry and current holdings of shares. They demonstrate the method in a simulation where the stock price process follows Geometric

Brownian motion, Eq. (3.1). Rebalance of the hedge is done five times a day.

They teach the agent with one-step SARSA. SARSA is very similar to Q-learning, but it is on-policy. The update rule for SARSA is (Sutton and Barto 2018)

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_{t+1})\right] \tag{3.20}$$

where the Q-value of the next state is taken from the action drawn from current policy, compared to Q-learning where the action is drawn from the optimal policy.

They teach the agent to hedge with a utility function, that has trading cost and variance trade off. The preference of the trade off is determined by the agent's risk-aversion parameter. The utility function to be maximized is

$$\max\left(E[w_T] - \frac{\kappa}{2}V[w_T]\right) \tag{3.21}$$

where w_T is the change in wealth during the episode and κ is a risk-aversion weight. They approximate this utility function with a reward function

$$R_t = \delta w_T - \frac{\kappa}{2}\left(\delta w_T\right)^2 \tag{3.22}$$

to translate it into a reinforcement learning problem. This approximation is criticized by J. Cao et al. (2019) for not being accurate when trading costs are high. Their results indicate a statistically significant improvement in reducing hedging costs and a small but non-significant increase in the variance of the agent's wealth.

J. Cao et al. (2019) consider reinforcement learning hedging in a stochastic volatility environment. They show the reinforcement learning hedging performance first in a similar setting to that of Ritter and Kolm where the asset price follows Geometric Brownian motion. In their second setting the volatility of the underlying asset is stochastic, but the volatility can not be hedged with another non-linear derivative instrument. The stochastic volatility is modelled by the SABR model by Hagan et al. (2002).

Their choice of reinforcement learning method is Q-learning, similarly to Halperin. They use similar observation of the state as Ritter and Kolm and use proportional transaction costs of 1 per cent while Ritter and Kolm had quadratic transaction costs. They prefer to use realized cash flows in the reward function instead of the change in the agent's wealth (*"accounting P&L"*) like Ritter and Kolm. Their reasoning is that the accounting P&L requires a pricing model whilst the cash flow method does not. Cash flows for each step are

$$CF_t = S_{t+1}\left(H_t - H_{t+1}\right) - \kappa|S_{t+1}\left(H_{t+1} - H_t\right)| \tag{3.23}$$

which is the amount of cash needed to increase or decrease the position in the underlying asset at mid price and the transaction costs of proportion κ. Their objective is to minimize the expected hedging costs C_t from time t onwards and to minimize the standard

deviation of future hedging costs. More formally this is

$$Y_t = \mathbb{E}(C_t) + c\sqrt{\mathbb{E}(C_t^2) - \mathbb{E}(C_t)^2} \tag{3.24}$$

Estimation of the future standard deviation is a challenge where they take a different approach to Ritter and Kolm. They introduce second Q-function to approximate $\mathbb{E}(C_t^2)$,

$$F(S_t, a) = Q_1(S_t, a) + c\sqrt{Q_2(S_t, a) - Q_1(S_t, a)^2} \tag{3.25}$$

for which Q_2 can be learned with Q_1.

They discretize the action space heavily, saying that the hedges are rounded to the nearest 10% of the underlying assets of the option. This essentially means that their hedge ratios can only take values with 0.1 intervals. Even with this simplification of the problem they show both lower mean hedging costs and lower standard deviation compared benchmarked with Black-Scholes delta hedging in the GBM setting. They show increasing advantage over delta hedging as rebalancing frequency increases, which is logical as the benchmark's transaction costs increase. In the stochastic volatility setting they show a large decrease in the hedging costs but a slight increase in the standard deviation.

4 PROPOSED MODEL

The reinforcement learning method implemented in this thesis is the TD3 introduced in chapter 2. Simpler methods suitable for continuous action spaces were considered and tested, such as vanilla actor critic methods and DDPG, but the complexity and stochasticity of the environment lead to learning problems. The methods seemed to either not converge at all, or started to learn and then later diverged. The stabilizing features introduced in TD3 seemed to make it the most viable method. This chapter will show the implementation details of the method and the general features of the hedging environment. The method will learn in three different settings, consisting of three different stock price processes: geometric Brownian motion, Heston process, and historical data of SPX options.

4.1 Hedging setting

The options considered in this thesis are vanilla European call options with time to expiry of less than 45 days. In the beginning of each episode we will choose an option which is at-the-money or as close as possible. These options hold some attractive properties. The delta of the option is known to move the most when the option is at-the-money i.e. the gamma of the option is at its highest. The larger the movements in the delta between each rebalancing step, the harder the option is to hedge in a market with frictions. When the option gets closer to its expiry the time value of the option decreases and the option delta changes even if the underlying asset stays put, known as the option charm or delta decay (Mastinsek 2012). Both Ritter and Kolm (2018) and Cao et al. (2019) consider at-the-money options. Ritter and Kolm consider options with expiry in 10 days while Cao et al. consider expiries of one month and three months.

These properties makes the relationship between the asset price, its volatility, time to expiry and the optimal hedge complex, which gives the agent a challenging MDP to learn. Exposures such as gamma cannot be hedged in similar fashion to delta hedging with holding a position in the underlying instrument, as hedging gamma exposure instead requires a position in another option or other derivative (Hull 2012). The agent can only trade in the underlying asset to hedge the option, and thus musts accept the gamma exposure. Neither can the agent hedge any volatility exposure, such as vega. Allowing hedging with only the underlying asset is similar to the earlier studies, except the method by Buehler, Gonon, Teichmann and Wood (2018) who allow hedging with other

derivatives.

The properties of the options considered here are in no way constraints to the hedging ability of the agent: the algorithm can be used to hedge a short or long position in a call or put option, the time to expiry can be longer or shorter than in this study, and the option can be as much out-of-money or in-the-money as one wants. With very simple modifications the algorithm could be extended to hedge exotic options as well.

As the agent hedges a long position in a call option, the position in the underlying asset will always be short. Similarly to the Black-Scholes assumptions unlimited short selling is allowed. Ritter and Kolm (2018) also hedge a long position while Cao et al. (2019) hedge a short position, but both use call options. Additionally, the underlying asset can be sold or bought in any fractions, and the tick sizes of the underlying asset are not considered. This combined with a continuous action RL method allows the agent to hedge on any precision it chooses to. In the methods introduced by Ritter and Kolm (2018) the hedging precision is 1% and in the work by Cao et al. (2019) the precision is 10 % of the underlying assets. The limited precision by Ritter and Kolm and Cao et al. is not an issue if the hedged position is small. If the option is written for example for 100 stocks, then a precision of 100 distinct actions, as used by Ritter and Kolm, is sufficient. If the position to be hedged grows to thousands or hundreds of thousands of options, having perfect precision with a discrete action method becomes impossible. The continuous action method works for any precision, and assuming convergence of policies should result in more optimal hedging.

We set a similar hedging objective function to the one that Ritter and Kolm (2018) used. The objective function is

$$Y = \mathbb{E}\left[\sum_{t=1}^{T} \mathsf{PnL}_t\right] - \kappa \mathbb{V}[\mathsf{PnL}] \tag{4.1}$$

where PnL_t (Profit-and-Loss) is the change in the agent's wealth between time steps. P&L for the episode can be expressed as the sum of all one time step P&Ls. The variance of the P&L is multiplied by a risk aversion parameter κ. Cao et al. (2019) used almost similar objective function, but used standard deviation instead of variance. The objective function holds some attractive properties. Large parts of the P&L is assumed to come from transaction costs. The objective function essentially becomes a trade off between transaction costs (expected P&L) and risk (the variance of the P&L). The risk aversion parameter makes it possible to find multiple, different policies where different weight is given to the elements.

The PnL_t, the profit or loss from time $t-1$ to t, is

$$\mathsf{PnL}_t = H_t^O \left(C_t - C_{t-1}\right) + H_t^S \left(S_t - S_{t-1}\right) - c|S_t \left(H_t^S - H_{t-1}^S\right)| \tag{4.2}$$

where H_t^O is the position in the option and H_t^S the holdings in the underlying asset at time t, C_t the price of the option and S_t the price of the underlying asset at time t, and c is the size of proportional transaction costs.

The objective function is approximated at each step by a reward function

$$R_t = \mathrm{PnL}_t - \kappa \left(\mathrm{PnL}_t\right)^2 \tag{4.3}$$

which is used in teaching the agent.

Increasing the steps in a day should make the algorithm hedge better, as stated by J. Cao et al. (2019): balancing the hedge more frequently is only a particular case of the first one. However with a neural network based policy and critic, convergence is not guaranteed. Choosing longer episode length would be natural for an RL agent which tries to maximize its lifetime rewards. However on the flip side, one needs to think about both the stock price process of the underlying asset and the market's option pricing process. If either one changes, then the environment in which the agent knows how to operate in no longer exists. Another factor is obviously the robustness and quickness of the training process which both need to be taken into account in a resource limited setting.

To quantify the hedging performance of the policy we will monitor it against benchmarks. The chosen benchmarks are the simple Black-Scholes delta hedge in Eq. (3.9) along with the Whalley-Wilmott hedge in Eq. (3.19).

4.2 State observation

To fulfil the Markov property the state observation must contain all information required to choose the optimal action. In practice, this requirement is often impossible to fill. The chosen features of the state observation are

- Moneyness of the option
- Time to maturity
- Current position in the underlying asset (the amount shorted)
- Black-Scholes implied volatility

It is worth noting that the Black-Scholes implied volatility of the option is a constant in the geometric Brownian motion setting, where volatility is constant. It is excluded in the GBM setting to boost training performance. In the Heston's stochastic volatility model the Black-Scholes implied volatility is stochastic, influenced by the instantaneous variance of Eq. (3.11), which is a CIR process. In the setting with empirical option price data the implied volatility is dependent on the observed option prices and thus time varying. The state observation does not include any data on the hedging performance so far. Performance of previous steps does not matter in choosing the optimal action, which is function of only the current state observation and the agent's estimate on the future states.

Alternative to having the moneyness of the option would be to include both the stock price and the strike price. Strike price would remain same for all steps in an episode and it provides no valuable information for the agent, which is why moneyness was chosen. If the agent is learning to hedge a single option, only the asset price would be enough.

In the settings considered in this thesis this is true. However for future improvements the moneyness enables the model to be generalized for multiple different options. Using moneyness is consistent with other Neural Network option pricing and hedging papers as pointed out by Ruf and Wang (2019). There are arguments for including the asset price: if one assumes non-rational markets where the option price is affected by the face value of the stock. Inclusion of historic stock prices would be relevant if one assumes that the investors are affected by the historic path of the asset price, but this kind of estimation is outside the scope of this thesis.

Ritter and Kolm (2018) used the asset's current price, time to maturity and the current position in the underlying asset as the state observation. They train their model to hedge a single option so the strike price is not relevant, as discussed above. Neither is any observation of volatility relevant as they use GBM for stock price process. They suggest adding option greeks to improve hedging performance, but do not include the greeks in their observation to see if RL can learn a hedging policy independently. Similar reasoning for leaving all greeks out was used here. We want to learn a policy independent of any pricing model. Cao et al. (2019) use the exact same state observation even for the stochastic volatility setting. Without any indication of the current volatility, or the markets' assumption of the current volatility, the Markov property is lost in a setting where the volatility can change drastically over time.

4.3 Implementation of the TD3 algorithm

By far the largest difference between the method introduced in this thesis and the methods introduced by Halperin (2017), Ritter and Kolm (2018) and Cao et al. (2019) is that the earlier works all use discrete action reinforcement learning, while we introduce a continuous action solution. Continuous actions allow for the higher hedging precision, as discussed earlier. The method chosen by Ritter and Kolm (2018) was one-step-SARSA, while both Halperin (2017) and Cao et al. (2019) used Q-learning.

The implementation of the TD3 algorithm is based on the paper that introduced TD3 (Fujimoto, Hoof and Meger 2018) and the original implementation (Fujimoto, Hoof and Meger 2020). For TD3 hyper parameters the Fujimoto et al.'s implementation used a tau of 0.005 and a policy update frequency of 2. Tau controls how fast the target networks are updated and policy update frequency controls how delayed the policy update is. Implementation in this thesis uses a tau of 0.001 for slower target network update and a policy update frequency of 2.

With a continuous action RL method with a stochastic policy noise should be added for action space exploration. TD3 has built in source for noise, as it adds noise to the targets. On top of that we add noise to the agents action from a normal distribution with standard deviation between 0.2 and 0.7. The standard deviation is high in the beginning for fast exploration and decreases when the performance increases. The hyper parameters were selected empirically by assessing both the performance and the speed of convergence

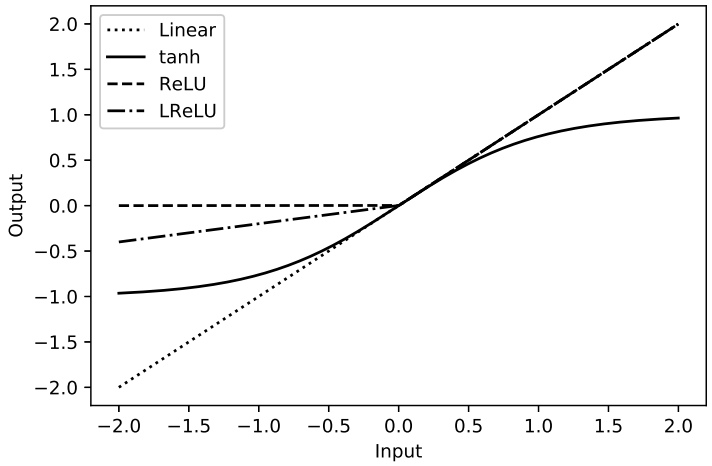

Figure 4.1. *Linear, tanh, ReLU and Leaky ReLU activation functions.*

of different parameter sets, and by selecting the one that performed sufficiently.

4.3.1 Actor and Critic Neural Networks

The original TD3 implementation uses relatively simple structures for the actor and critic networks. Both the actor and critic consist of two hidden layers of size 256 and the activation function used in the paper is Rectified Linear Unit ("ReLU"). We add another hidden layer for a total of three hidden layers for both the actor and critic. Layer size is close to the original, 250 for each hidden layer. The difference of adding another layer seemed to provide small improvement on the performance. For activation functions for the hidden layers we change ReLU to Leaky ReLU. Leaky ReLU allows a small leakage for negative values (Maas, Hannun and Ng 2013). The amount of leakage is controlled by a hyper parameter α:

$$h(i) = \begin{cases} i & \text{if } i > 0 \\ \alpha \times i & \text{otherwise.} \end{cases} \tag{4.4}$$

When $\alpha = 0.0$ the function becomes ReLU. Common values for the leakyness parameter are between 0.05 and 0.10. (Maas, Hannun and Ng 2013) Leaky ReLU was chosen, as it has been shown to improve neural networks' performance (B. Xu et al. 2015). Empirical tests conducted suggested that Leaky ReLU works better than ReLU, and that 0.10 is suitable for the leakyness parameter.

The activation function for the actor's output layer is hyperbolic tangent. Hyperbolic tangent outputs a value from a range of [-1, 1] which is later scaled outside the neural network to a range of [0, 1] which is the limited position the agent is allowed to take in the underlying asset. For the critic network the output layer outputs a single value with linear activation. An illustration of the three used activation functions together with ReLU is seen in figure 4.1.

The learning rate in both the actor and critic network is relatively small, 1e-5. Together with a small learning rate and a high batch size of 10,000 a stabilizing effect is achieved. The batch size of 10,000 is higher than usually used in TD3 or DDPG implementations, where the usual batch size is between 64 and 256 (Fujimoto, Hoof and Meger 2018; Lillicrap et al. 2015). Batch sizes of this size are not unheard of for continuous action problems, and used regularly in algorithms like TRPO by Schulman et al. (2015). Adam is used as the optimizer.

After each Leaky ReLU activation we use dropout activation. Dropout was introduced by Hinton et al. (2012) and is used to overcome overfitting of neural networks. The use of dropout layers has received very little attention in the reinforcement learning literature. The reason for little attention might be that overfitting is often a desired outcome in reinforcement learning, contrary to most supervised learning tasks where overfitting is avoided at all costs. Overfitting is less of a problem when the stochasticity of the environment is small: single samples can give a good picture of the optimal action in a state.

In this study it was seen that without dropout layers the critic networks started to overfit after several thousands of episodes, which resulted in a diverging policy. The overfitting might have been due to the fact that same samples were drawn from the replay buffer over and over again. The p-value used for the dropout layers was seen to have a significant effect on the hedging performance. The original paper suggests a p-value of 0.5, but that was seen as too high for this problem. A p-value of 0.15 was found to improve the results the most, and was used in this thesis. The optimal p-value is of course dependent on the other hyper parameters such as the network layer sizes.

4.3.2 State featurization

Usage of a featurization function is a known trick to improve the performance of neural networks. Featurization means running the input data through a featurization function engineered to a specific problem. Featurization can be used to improve the ability of neural networks to learn the underlying patterns (Ekenel and Stiefelhagen 2006). Polynomial, interaction and normalization techniques were used in this thesis to improve the hedging performance.

Polynomial featurization means taking a n-th degree polynomial from a chosen feature or features. Polynomial features can be used if a feature is suspected to have polynomial correlation to the desired output. For the state observation we ran only polynomial featurization for the moneyness, which was raised to the power of two. The original moneyness was also kept. Interaction features are interactions between two or more features, e.g. multiplication, division or exponential interaction. Here a multiplication of moneyness and the implied volatility was used. The chosen polynomial and interaction features seemed to have a small positive effect on the hedging performance.

Normalization, or standardization, means calculating the z-score for each feature in the

input vector. Z-score is calculated by subtracting the feature mean and dividing by the standard deviation

$$z_{i,n} = \frac{x_{i,n} - \overline{x}_i}{s_i} \tag{4.5}$$

for each feature *i* in the input vector *n*. Standardization has been shown to increase the performance of neural networks (Ekenel and Stiefelhagen 2006), by improving the predicting accuracy or reducing learning times, often achieving both. The importance of standardization increases if the feature vector contains features which have a different order of magnitude. A feature with higher order of magnitude might dominate other features without standardization. The normalization is implemented with the skicit-learn library (Pedregosa et al. 2011).

5 EXPERIMENTAL SETUP AND RESULTS

The method is tested on three different option price processes: geometric Brownian motion, Heston's stochastic volatility model and historical SPX option and underlying prices. All three settings are tested with varying risk aversion parameter and transaction costs. All tests displayed are out of sample simulations.

The agent introduced in the previous chapter will be tested in three different settings. In this chapter the details of the settings will be introduced and the agent's performance in all three will be shown. First we will evaluate the agent in a simple environment where the volatility is constant and the stock price follows geometric Brownian motion. In the second environment we will see how the agent deals with stochastic volatility as given by the Heston model. Finally, we will use a simplified version of transfer learning to train the agent in a Heston model-simulated environment and test with real SPX option data.

5.1 Geometric Brownian motion

The first environment we test the agent in is the closest to satisfy the Black-Scholes assumptions described in chapter 3. We set a constant volatility, $\sigma = 25\%$. For the constant risk free rate we set $r = 1.2\%$. The underlying asset price follows geometric Brownian motion as seen in equation (3.1), with a drift $\mu = 0.0\%$. The Black-Scholes model expects continuous time hedge re-balancing, but we set the discrete time interval to one hour. This equals seven balances a day. Episode length will be set to 35 time steps, five trading days with seven steps a day. Ritter and Kolm (2018) used episode lengths of 10 days with 5 rebalances a day. J. Cao et al. (2019) used longer hedging periods, one and three months. Their rebalance frequence was substantially longer, from daily to weekly.

We test the agent and compare to benchmarks with different transaction costs. We start with no transaction costs ($c = 0.0\%$), continue with moderate transaction costs ($c = 0.1\%$) and end with high transaction costs ($c = 1.0\%$). For all settings we will use the Black-Scholes delta hedge in Eq. (3.9) and the Whalley-Wilmott hedge in Eq. (3.19) as benchmarks. For the Whalley-Wilmott hedge we set the risk aversion parameter to 5. We use 10,000 simulated stock price paths for each test.

To assess the hedging performance of the agent and the benchmarks, we will consider four values from the tests. The mean episode P&L is the average of the total P&L over each five day episode. The episode P&L standard deviation is the standard deviation

Table 5.1. *Reinforcement learning agent's hedging cost and performance against benchmarks, Black-Scholes delta hedge ("BS") and Whalley-Wilmott hedge ("WW"). The hedge period is 5 days and the option is a call option expiring in two weeks. Hedge is rebalanced seven times a day. The stock price process follows geometric Brownian motion. There are no transaction costs ($c = 0.0\%$). The risk aversion parameter is high ($\kappa = 1.5$). Tested on 10,000 out-of-sample simulations.*

	Agent	**BS**	**WW**
Mean episode P&L	-1.904	-1.907	-1.907
Time step P&L std	0.372	0.370	0.370
Episode P&L std	1.224	1.217	1.217
Rewards	-4.660	-4.620	-4.620

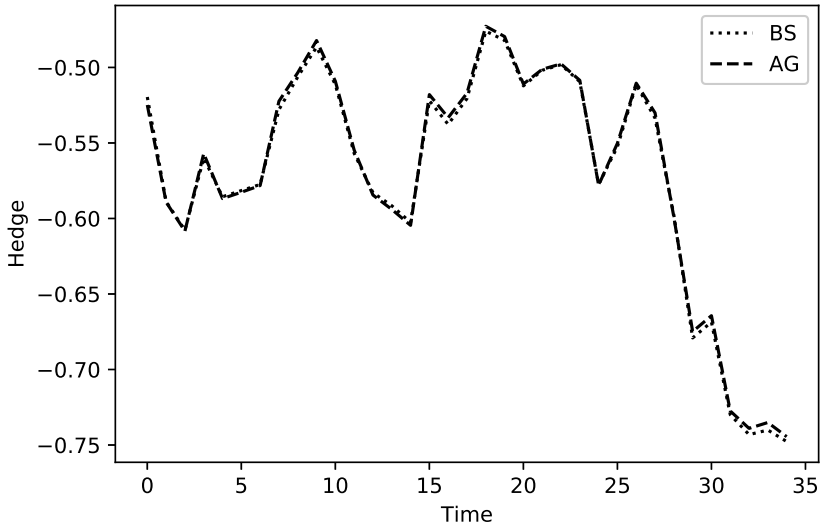

Figure 5.1. *An example of Agent's hedging policy compared to the benchmarks in the zero transaction costs scenario.*

of the episode total P&Ls over the 10,000 tests. The time step P&L standard deviation is the standard deviation of P&L between single rebalances. Finally, we will monitor the average rewards accumulated, as determined in Eq. (4.3).

5.1.1 No transaction costs

Fist we will consider a special setting where there are no transaction costs. An optimal hedge in a continuous time GBM-world would be the Black-Scholes delta hedge. With discrete time steps, the optimal hedge might differ a bit. We set the risk aversion parameter κ to 1.5. One should note that the Whalley-Wilmott benchmark hedge equals Black-Scholes hedge when transaction costs are zero. The results are displayed in table 5.1. The rewards is the mean of rewards received in an episode, and should be viewed as an indicator of which policy is the best at fulfilling the objective function. We see

Table 5.2. *Reinforcement learning Agent's hedging cost and performance against benchmarks, Black-Scholes delta hedge ("BS") and Whalley-Wilmott hedge ("WW"). The hedge period is 5 days and the option is a call option expiring in two weeks. Hedge is rebalanced seven times a day. The stock price process follows geometric Brownian motion. Moderate transaction costs ($c = 0.1\%$). The risk aversion parameter is low ($\kappa = 0.5$) in the first test and high ($\kappa = 1.5$) in the second. Tested on 10,000 out-of-sample simulations.*

$\kappa = 0.5$	**Agent**	**BS**	**WW**
Mean episode P&L	-3.580	-3.777	-3.236
Time step P&L std	0.386	0.383	0.404
Episode P&L std	1.151	1.099	1.337
Rewards	-5.130	-5.200	-5.380
$\kappa = 1.5$	**Agent**	**BS**	**WW**
Mean episode P&L	-3.654	-3.765	-3.185
Time step P&L std	0.384	0.383	0.405
Episode P&L std	1.156	1.110	1.379
Rewards	-8.280	-8.330	-9.440

that the agent learns a policy that almost matches the benchmark performance, but has slightly higher standard deviation. The agent learns a policy that very closely replicates the Black-Scholes delta hedge, as can be seen in figure 5.1.

5.1.2 Moderate transaction costs

In the moderate transaction costs setting we will considerate reasonable transaction costs of 0.1 %. We set the risk aversion parameter first to 0.5 and then to 1.5. The results of the tests are displayed in table 5.2 for both 0.5 and 1.5 risk aversion parameter. We can see that the agent learns to satisfy the objective function slightly better than its benchmarks. The agent has lower average costs than the BS hedge but higher costs than the WW hedge. The agent is also between BS and WW in the standard deviation, slightly losing to BS hedge. The agent sacrifices a bit in the standard deviation to save in costs compared to BS, but WW sacrifices even more. An example of the difference three hedges is displayed in figure 5.2, which corresponds to the setting in table 5.2 with risk aversion parameter of 0.5.

The difference of 1.0 in the risk aversion parameter κ does not seem to have a high effect in the moderate transaction cost setting. The are small differences in the mean P&L and standard deviation. However, the differences are of similar size than the performance differences between different runs of the algorithm.

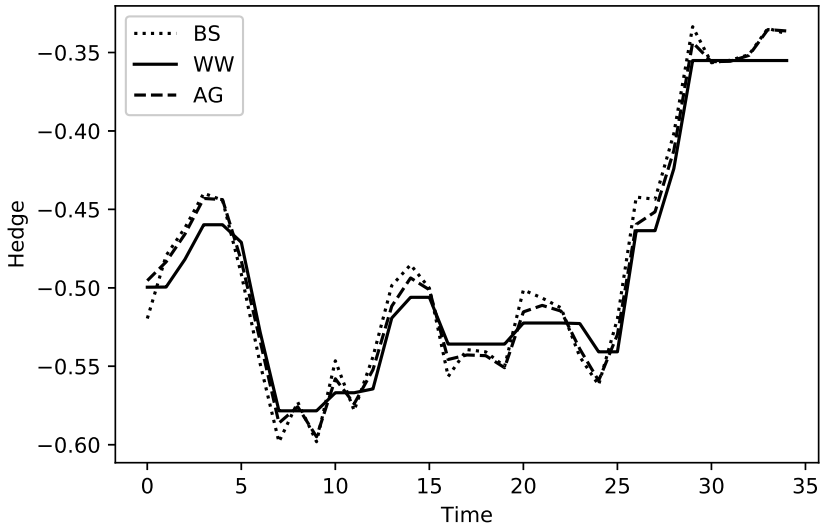

Figure 5.2. *An example of Agent's hedging policy compared to the benchmarks in the moderate transaction costs scenario.*

Table 5.3. *Reinforcement learning Agent's hedging cost and performance against benchmarks, Black-Scholes delta hedge ("BS") and Whalley-Wilmott hedge ("WW"). The hedge period is 5 days and the option is a call option expiring in two weeks. Hedge is rebalanced seven times a day. The stock price process follows geometric Brownian motion. High transaction costs ($c = 1.0\%$). The risk aversion parameter is high ($\kappa = 1.5$). Tested on 10,000 out-of-sample simulations.*

	Agent	**BS**	**WW**
Mean episode P&L	-8.919	-18.55	-9.838
Time step P&L std	1.171	1.207	1.177
Episode P&L std	5.415	2.910	2.538
Rewards	-42.19	-54.90	-43.38

5.1.3 High transaction costs

Finally we consider high transaction costs of 1.0 % for the GBM setting. The risk aversion parameter is set to 1.5. The results are displayed in table 5.3. We see that the agent learns a policy that has the highest mean P&L, compared to both benchmarks. The standard deviation between each time steps is also lower than either of the benchmarks. However, the standard deviation between episodes is higher than the benchmarks have, meaning that agent has both more good weeks and bad weeks compared to the average.

The hedging policy learned in the high transaction costs scenario is a very interesting one. The agent learns to avoid the transaction costs by being very hesitant to change the hedge too rapidly. It favours waiting and seeing if being under or over hedged would pay out. This is seen in figure 5.3. We see that the BS hedge changes its position most often while the WW hedge avoids transactions costs but stays close to the BS hedge. The

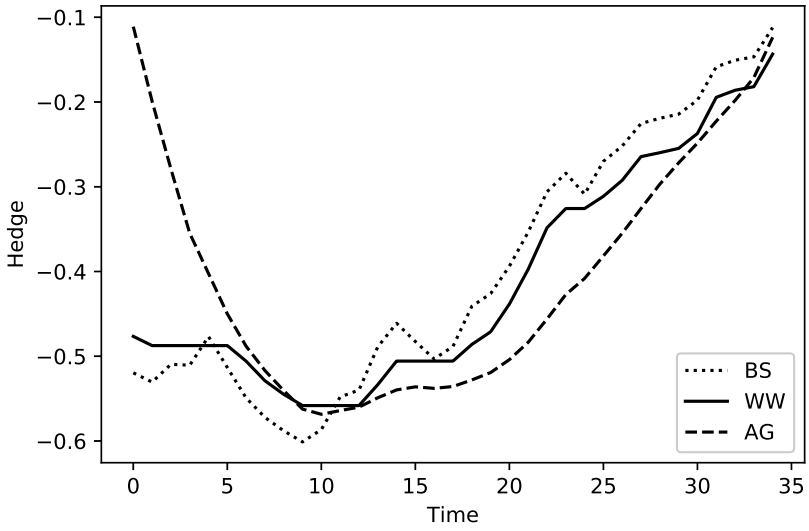

Figure 5.3. *An example of Agent's hedging policy compared to the benchmarks in the high transaction costs scenario.*

agent however has a different approach. Setting the initial hedge is very costly, as at time $t = 0$ the holdings are zero, and the agent prefers to set up the hedge slowly. Similarly the agent is slower to decrease the hedge. This behaviour is perfectly explained by the reward function in Eq. (4.3). Large sudden transaction costs would cause a negative effect on both components of the reward function. From the high episode P&L standard deviation we see that the policy sometimes results in a very good P&L and sometimes it loses a lot, as the agent is under hedged in the beginning of the episode. When the transaction costs are high, the approximation of the objective function Eq. (4.1) by the reward function (4.3) does not seem to hold, as the mean P&L diverges from zero. This fault in the reward function was pointed out by J. Cao et al. (2019).

5.2 Heston model of stochastic volatility

Next we will consider an environment of stochastic volatility, where the asset price follows Eq. (3.10) and the volatility follows Eq. (3.11). For the simulations we set the long time average variance to equal the constant volatility of 25% in the GBM setting: $\theta = 0.0625$. Other parameters are $\rho = -0.5, \kappa = 2, \epsilon = 0.4$ and $v_t = 0.04$ (20% volatility equivalent). For the constant risk free rate we set 1.2% and for the stock price drift we set 0%. The hedge rebalance interval and the length of an episode is same as in the GBM setting, 1 hour and 35 time steps.

The benchmarks will again be Black-Scholes delta hedge and Whalley-Wilmott hedge. For both the volatility used is the implied volatility derived from the price calculated with the Heston model in Eq. (3.13). We will first consider a scenario with no transaction costs to see if the agent is able to outperform its benchmark in minimizing the P&L standard deviation. After that we will consider a setting with transaction costs of 0.1 %. We use

Table 5.4. *Reinforcement learning Agent's hedging cost and performance against benchmarks, Black-Scholes delta hedge ("BS") and Whalley-Wilmott hedge ("WW"). The hedge period is 5 days and the option is a call option expiring in two weeks. Hedge is rebalanced seven times a day. The stock price process follows Heston's model with stochastic volatility. There are no transaction costs ($c = 0.0\%$). The risk aversion parameter is high ($\kappa = 1.5$). Tested on 10,000 out-of-sample simulations.*

	Agent	**BS**	**WW**
Mean episode P&L	0.093	0.067	0.067
Time step P&L std	0.579	0.602	0.602
Episode P&L std	2.501	2.727	2.727
Rewards	-8.740	-9.470	-9.470

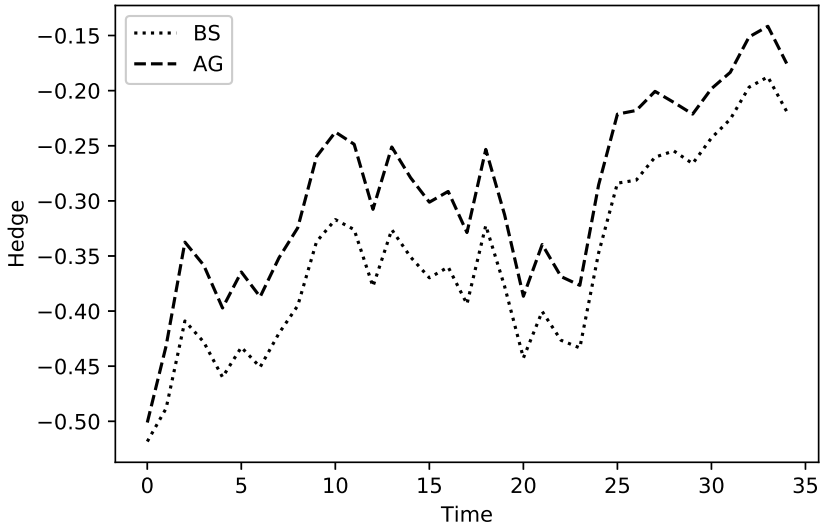

Figure 5.4. *An example of Agent's hedging policy compared to the benchmarks in the zero transaction costs and stochastic volatility scenario.*

10,000 simulated stock price paths for each test.

5.2.1 No transaction costs

For the no transaction costs scenario we will set the risk aversion parameter to 1.5. The results are displayed in table 5.4. From the results we see that the agent clearly outperforms its benchmarks, by achieving a lower standard deviation with slightly higher P&L. The agent's hedging policy and BS hedge is illustrated in figure 5.4. The difference to the GBM no transaction costs setting in figure 5.1 is clear. When the stock price followed GBM the agent learned to replicate the BS delta almost perfectly. Here there is a clear difference between the Agent's hedge and BS hedge, which is natural as BS hedge is not designed for the stochastic volatility.

Table 5.5. *Reinforcement learning Agent's hedging cost and performance against benchmarks, Black-Scholes delta hedge ("BS") and Whalley-Wilmott hedge ("WW"). The hedge period is 5 days and the option is a call option expiring in two weeks. Hedge is rebalanced seven times a day. The stock price process follows Heston's model with stochastic volatility. Moderate transaction costs ($c = 0.1\%$). The risk aversion parameter is low ($\kappa = 0.5$) in the first test and high ($\kappa = 1.5$) in the second. Tested on 10,000 out-of-sample simulations.*

$\kappa = 0.5$	**Agent**	**BS**	**WW**
Mean episode P&L	-1.631	-1.778	-1.162
Time step P&L std	0.588	0.608	0.629
Episode P&L std	2.460	2.718	2.854
Rewards	-3.860	-4.150	-4.050
$\kappa = 1.5$	**Agent**	**BS**	**WW**
Mean episode P&L	-1.716	-1.792	-1.166
Time step P&L std	0.580	0.608	0.628
Episode P&L std	2.409	2.736	2.874
Rewards	-9.740	-10.65	-10.97

5.2.2 Moderate transaction costs

For moderate transaction costs we set the transaction costs to 0.1% of the face value of each transaction. We test with risk aversion parameters of 0.5 and 1.5. The results are displayed in table 5.5. Similarly to the no transaction costs scenario, we see that the agent's hedging performance is better than the benchmarks'. Again the Whalley-Wilmott method provides the lowest transaction costs but with higher standard deviation. The Black-Scholes hedge has both higher transaction costs and higher standard deviation. Between the policies with different risk aversion parameters a higher difference is seen than in the GBM setting. A sample of the difference in the hedging policies is seen in figure 5.5.

5.3 Empirical data, SPX options from 2012 and 2013

Simulations are attractive proxies for training agents intended for real life usage. Simulations can be ran as many times as needed, and they provide a safe environment for training where mistakes do not matter. Training a robot or a hedging agent outside of simulation, in real life, could end up being very expensive. However, the policy an agent learns is often specific to the simulated environment, and a problem of how to transfer the learning to real life arises. Sim-to-Real reinforcement learning studies the methods of transferring the agent's policy from simulation to real life. (Peng et al. 2017)

We will apply the most basic form of Sim-to-Real reinforcement learning. The agent will be trained on a simulated setting and tested on historical data. The simulations are equiv-

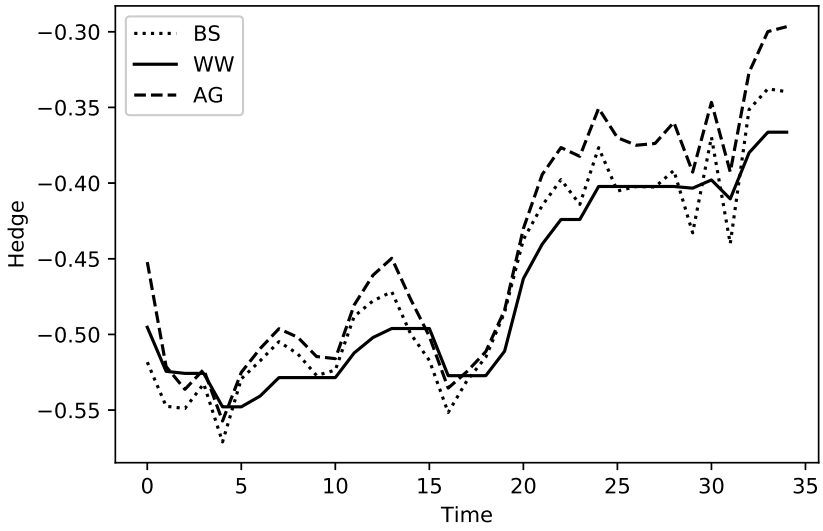

Figure 5.5. *An example of the agent's hedging policy compared to the benchmarks in the moderate transaction costs and stochastic volatility scenario.*

alent to the stochastic volatility setting described above. For each week's simulations, we will calibrate the Heston model to the starting point of the week, and use the calibrated Heston model for the simulated SPX and option prices. Each week is simulated for at least 5,000 episodes or until convergence on simulated data is reached. This means that a hypothesis is made that the Heston model, calibrated at the start of the week, can simulate the pricing well enough that the agent can learn a policy for the week.

Cao et al. (2019) suggested training an agent on multiple stock price processes if the decision maker is unsure which process models the asset price the best. Similar idea is used by Peng et al. (2017). Adding randomness, e.g. different stochastic stock price processes, during training enables the agent to learn a policy that tolerates the uncertainty of reality. Uncertainty is certainly a feature that describes the real stock price paths.

We assume that the investor can buy and sell the SPX index itself, which is not possible in reality. Real life applications could hedge SPX options with any instruments that track the index, e.g. exchange traded funds. This is an accepted proxy in case of SPX option hedges, and has been used in earlier studies (e.g. Bakshi, C. Cao and Chen 1997).

We use the data of SPX options from 3 January 2012 to 31 December 2013. The data set contains the option bid and ask prices and the underlying price, which will be used for hedging. The P&L is calculated from the mid price of the option. The roots considered are SPX, the monthly options expiring during the night after the third Friday. For each day we take data each hour starting from 9:31 until 15:31 for a total of seven data points per day. This equals the seven rebalances in the simulations and settings described earlier. The combined lenght of the hedging periods are 3491 time steps or 499 trading days.

Again options closest to at-the-money as possible are considered. The options chosen at the start of each five trading day hedging period always have expiry after the hedging

Table 5.6. *Reinforcement learning Agent's hedging cost and performance against benchmarks, Black-Scholes delta hedge ("BS") and Whalley-Wilmott hedge ("WW"). The hedge period is 5 days and the option is a call option with expiries ranging from 5 days to one month. Hedge is rebalanced seven times a day. The agent is taught by simulating the stock price paths with Heston's model with stochastic volatility. Moderate transaction costs ($c = 0.1\%$). The risk aversion parameter is low ($\kappa = 0.5$) in the first test and high ($\kappa = 1.5$) in the second. Empirical data from 3 January 2012 to 31 December 2013. Options are at-the-money SPX options. Hedges are proxied with SPX index.*

$\kappa = 0.5$	**Agent**	**BS**	**WW**
Mean P&L per week	-2.017	-2.474	-1.667
Time step P&L std	0.704	0.692	0.717
Week total P&L std	2.360	2.400	2.519
Rewards	-5.451	-5.553	-5.428
$\kappa = 1.5$	**Agent**	**BS**	**WW**
Mean P&L per week	-2.217	-2.474	-1.667
Time step P&L std	0.697	0.692	0.717
Week total P&L std	2.503	2.400	2.519
Rewards	-14.04	-14.01	-14.44

period ends. This limit is chosen to ensure that none of the hedging periods is cut short due to the option maturing and that the option being hedged does not change during the period. This means that the option which is being hedged can change every week, if the expiry comes too close or if another option is closer to at-the-money. These choices result to a set of options which have an average time to expiry of 19 days. At shortest the expiry at the end of a week is 2 days and at longest at the start of a week 40 days. The moneyness at the start of the hedging periods ranges from 99.77% to 100.16%, with the average being 100.0% as intended.

The risk free rate used is the 1 year point of the U.S. Treasury Yield Curve. The risk free rate is obtained from U.S. Department of the Treasury (2020). Heston model calibration is done by calculating a volatility surface from close to at-the-money options for multiple maturities. Strike-expiry combinations are taken by first limiting to options where the strike price is divisible by 50 and then taking the closest to at-the-money option. Then 50 and 100 points higher and lower strikes are added for a total of 5 strike prices. Black-Scholes implied volatility is then calculated from the mid price for each strike price and expiry combination. Finally, QuantLib (2020) is used to calibrate the model.

5.3.1 Test Results

In the back testing setting we will set transaction costs to 0.1%, same as in the moderate transaction costs setup with the simulated GBM and Heston stock price processes. We will train two agents, with risk aversion parameters set to 0.5 and 1.5.

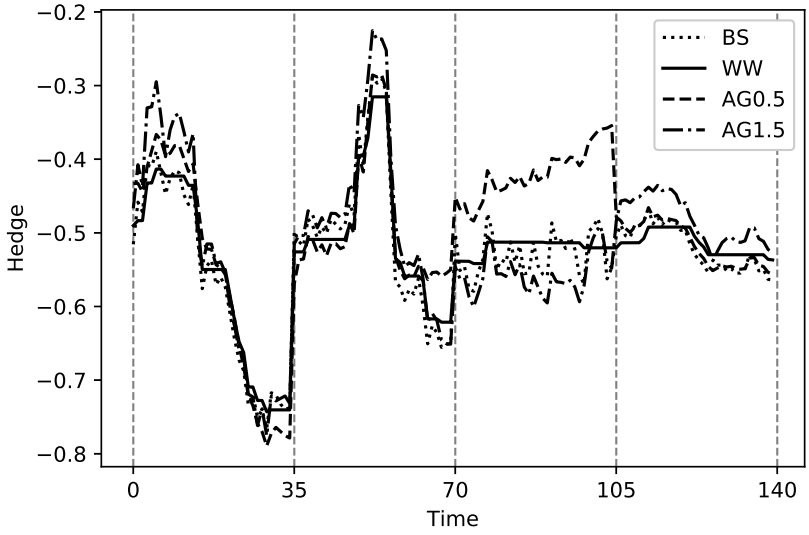

Figure 5.6. *An example of the agents' hedging policies compared to the benchmarks with empirical data of SPX options in the four weeks between 24 July 2012 and 20 August 2012. The period is 120 consecutive time steps. Transaction costs are 0.1%. AG0.5 is the agent with* $\kappa = 0.5$ *and AG1.5 is the agent with* $\kappa = 1.5$*.*

The results over the 100 week period are displayed in table 5.6. The P&L for both of the agents is higher than the Black-Scholes delta hedge's P&L. As the P&L is higher, a slight increase in the standard deviation is seen. Similarly to the simulated settings, the Whalley-Wilmott method had the highest P&L but also the highest standard deviation.

When considering single weeks, the low risk aversion agent satisfies the objective function better than the Black-Scholes in 57 out of 100 weeks and better than Whalley-Wilmott 50 times. The higher risk aversion agent satisfies the objective function better than BS 55 times and better than WW 58 times.

In most of the hedging periods both agents' follow roughly the positions taken by the benchmarks, Black-Scholes and Whalley-Wilmott, as seen in figure 5.6 in the first two and last week displayed. However, during some weeks there is a clear difference between the position the agents' take to the benchmarks, as seen in the third week. During that week the agent with a 0.5 risk aversion hedges less than others. There are a couple of weeks like this for both agents, where they show poor performance compared to the average. Low performance during some weeks might be caused by multiple reasons. One might be that the Heston model is poorly calibrated during the start of the week. Other obvious reason might be the little training time used for each week. Only 5,000 episodes were simulated for each week. Some sets of parameters for the Heston model might require more episodes for the agent to learn the optimal policy. Indications of this kind of behaviour was noticed during training of the model. For some weeks the policy converged very soon, after 1,000 episodes. For some weeks, 5,000 episodes was not enough. Simulating 5,000 episodes took approximately 15 minutes on a high end commercial GPU, and thus training two agents for 100 weeks took 50 hours. In

applications to shorter time periods more time could be spent to ensure correct model calibration for simulation and more training episodes for each week.

The results indicate that the method is transferable from simulated environment to reality. The agent is able to learn a policy that work on both simulated and empirical data. The agent learns to avoid unnecessary transaction costs while making sure that the variance does not grow too much. Changes in the risk aversion parameter seem to have a desired effect on the policy.

5.4 Summary of the Results

The agents were trained with two different risk aversion parameters, $\kappa = 0.5$ and $\kappa = 1.5$, to learn two distinct policies. We saw that the risk aversion parameter has an expected effect on the hedging policy in the stochastic volatility simulation and empirical testing, but only a small effect in the GBM setting. The lower risk aversion parameter agent had lower hedging costs than the high risk aversion agent in all test where both were considered. Two different stock price processes were considered, geometric Brownian motion in Eq. (3.1) where the volatility is constant and Heston's model with stochastic volatility in Eq. (3.10).

The zero transaction cost setting where the stock price process follows geometric Brownian motion is the closest one to the Black-Scholes world, with the major difference being discrete time hedge rebalances compared to continuous rebalancing in the Black-Scholes setting. We saw that the agent learns a policy that replicates the Black-Scholes delta hedge. The results could be different if the rebalance interval was longer than the one hour used. With longer interval the variance minimizing position in the underlying asset could diverge from the position of Black-Scholes delta. The moderate transaction costs setting with geometric Brownian motion extended the Black-Scholes world by introducing transaction costs. We saw the the agent learned a policy that somewhat followed the Black-Scholes delta hedge, achieving higher P&L than BS hedge but lower than WW hedge.

In the high transaction costs scenario, on average, the agent is able to minimize the transaction costs better than its benchmarks. The standard deviation between the episodes is high, and the agent's policy of avoiding hedging sometimes pays out, but sometimes comes costly. The agent avoids setting the initial hedge due to the nature of the reward function. The approximation for the agent's reward function of the variance in the objective function does not hold when transaction cost are high.

In the stochastic volatility world of Heston we tested the agents with no transaction costs and moderate transaction costs of 0.1%. As expected, we see that the agent outperforms the benchmarks in these settings, measured by the reward function. The benchmarks are analytical solutions for a constant volatility setting, and are out of their depth in a world with stochastic volatility. In the moderate transaction costs setting, even though the agent fulfills the objective function better than the Whalley-Wilmott, the agent's P&L is still quite

a lot higher. It would be an interesting extension to test if the agent is able to learn a policy that beats the Whalley-Wilmott method if the risk aversion parameter was lower, and how much would the variance increase in that policy.

In the final setting the agent was taught in a simulation equivalent to the stochastic volatility setting. The agents policy was then tested on empirical option data, in an attempt to transfer the agent from simulation to reality. This method assumes that the Heston model can describe the real stock price process on a reasonable level.

We saw that, on average, transferring the agent's policy from simulation to reality provided sufficient hedging performance. The agent outperforms both benchmarks in more than 50% of the weeks. Over the whole two year period covering years 2012 and 2013, the agent is able to hedge with higher P&L than the BS benchmark, but similarly to the simulations the WW hedge had even higher P&L. Measured with the reward function, there were no drastic differences in the hedging performance. In some 5-day-periods in the empirical tests, the agent performed poorly and had problems converging to an optimal policy. Minimizing the occurrence of this problem could be achieved by increasing the number of episodes the agent is trained on or by introducing another stock price process.

6 CONCLUSION

This thesis examined the use of continuous action reinforcement learning in hedging of options. Both simulated and historical stock price paths and option prices were considered. The hedging problem was defined with an utility function, a method familiar from previous literature of both analytical and machine learning methods. The utility function considers both the expected P&L and the variance of the P&L, where the importance of the variance is controlled with a risk aversion parameter. The proposed method for teaching the agent to maximize the objective function is an application of the Twin Delayed Deep Deterministic Policy Gradient method by Fujimoto, Hoof and Meger (2018). An extensive but not exhaustive hyper parameter search was conducted to tune the algorithm for the problem. All the implementation details were presented for reproducibility.

The first research question was to assess the proposed methods' performance against benchmarks on simulated asset price paths. We trained and tested the reinforcement learning agent on two different stock price processes, the geometric Brownian motion and the stochastic volatility model of Heston. The performance of the proposed method was sufficient, outperforming the benchmarks on some settings and achieving equal or close to equal performance on others. It was noted that the approximation of the variance in the reward function does not hold when the transaction costs are high.

The second research question was to assess the performance on empirical data. A Sim-to-Real transfer learning method was proposed for applications on empirical data. The simulation and teaching of the agent was conducted on asset price paths simulated with the Heston model. The Heston model was calibrated for each week, and then 5,000 simulations were used to teach the agent, before testing on the actual option and underlying prices for the week. The data set used was SPX option and index prices from years 2012 and 2013. The proposed method is tied to the ability of the Heston model to simulate the stock prices process. The results show that the agent was able to learn a viable hedging policy from training in a simulated environment, but that the calibration of the Heston model is important for the training.

In the testing with empirical data, the agent had a few weeks where the performance was sub-par compared to the average and the benchmarks. The agent had a hard time converging with some calibrated set of parameters for the Heston model simulation. A solution for this in addition to better model calibration could be to use more episodes in the training period.

Real life applications could utilize a mixed method of choosing a long episode and retraining the agent to capture the changes in the environment. For example, the agent could be trained for five day periods as in this thesis, but retrained after a single day or even an hour. This would capture both the wish to optimize hedging for longer periods of time and the constantly changing environment often seen in the market.

The method could be paired with a traditional risk measure, such as VaR, to control for cases where the agent faces a situation that it has not experienced during training. The risk of facing these kinds of situation depends on how well the chosen simulation method describes the real stock price process. If the fit is poor, the agent could end up making costly mistakes. To avoid this, additional controls should be implemented. Another way to stabilize the learning and to counter the problems in capturing and calibrating the simulations would be to use multiple stock price processes for teaching the agent in the simulation phase. This was also suggested by J. Cao et al. (2019).

We saw that the transaction costs of the agent were often higher than those of the Whalley-Wilmott method, which was used as a benchmark. Although the agent was better at maximizing the utility function at least on some settings, the P&L of the agent was lower than WW's. Lower risk aversion parameters should be considered to see if the agent is able to learn a policy with very minimal transaction costs, and if that would create a large trade off in the standard deviation.

Lately, a lot of attention in the research concerning stock prices has been given to jumps. The stock price processes considered here do not model jumps. Training the agent with stock price processes that model jumps could improve the hedging performance for applications on empirical data or practical use. The empirical data used in this thesis is a relatively small and homogeneous sample. The years 2012 and 2013 can be considered to be years of relatively stable growth. Potential issues regarding times of distressed markets were not covered. The policy learned by the agent might be over or under hedged given drastic market moves. Or it might be optimally hedged even in a distressed scenario. To give an answer to this, more research is needed.

One of the limitations of the proposed method is the applicability to high transaction costs. In addition, further research should be done to study more how the method performs with different transactions costs. Even if this thesis succeeded in testing with three different transaction costs, more exhaustive tests should be performed. In addition, only linear transaction costs were considered. When the position to be hedged is large, transactions are bound to move the market. When hedging options written on illiquid instruments, even small transactions can have drastic effects on the instrument's price. Thus, the results here are only applicable for positions that have negligible effect on the market prices. Optimizing the hedging in very illiquid markets poses an interesting possibility for future research, where reinforcement learning could surely be utilized, but where challenges would arise from modeling the effects of transactions sufficiently.

Further research in the spirit of this thesis could be conducted by applying different con-

tinuous action reinforcement learning methods for a similar hedging problem setup. The selection of reinforcement learning algorithms are numerous and the field is ever evolving. This thesis used Sim-to-Real to test the agent on empirical data: the agent was trained on simulated data and then tested on real data. To extend this work, the agent could be trained on actual historical data. This could allow the agent to find hedging strategies independent of any simulated stock price process and would solve multiple problems. No calibration of the model or choosing between different stock price processes would be needed. The agent could learn patterns in the pricing that are not captured by the pricing models.

The contributions of this thesis in the field of hedging are following. This thesis builds on earlier work of reinforcement learning applications to hedging with utility functions. Earlier work has focused on discrete action spaces. As far as we are aware, this is the first study to apply continuous action reinforcement learning to hedging problem defined as a mean-variance problem. Similarly, as far as we are aware, no prior research has been conducted that shows the performance of reinforcement learning hedging on real data. The design choices and implementation details were described in detail. By following the documented implementation details any further research should be able to replicate the proposed method.

REFERENCES

Albrecher, H., Mayer, P., Schoutens, W. and Tistaert, J. (2007). The Little Heston Trap. *Wilmott*, 83–92.

Bakshi, G., Cao, C. and Chen, Z. (1997). Empirical Performance of Alternative Option Pricing Models. *The Journal of Finance* 52.5, 2003–2049. DOI: 10.1111/j.1540-6261.1997.tb02749.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1997.tb02749.x.

Barles, G. and Soner, H. M. (1998). Option pricing with transaction costs and a nonlinear Black-Scholes equation. *Finance and Stochastics* 2, 369–397.

Black, F. and Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy* 81.3, 637–654.

Buehler, H., Gonon, L., Teichmann, J. and Wood, B. (2018). Deep Hedging. *Quantitative Finance* 19.8, 1271–1291.

Buehler, H., Gonon, L., Teichmann, J., Wood, B., Mohan, B. and Kochems, J. (Jan. 2019). Deep Hedging: Hedging Derivatives Under Generic Market Frictions Using Reinforcement Learning. *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3355706.

Cao, J., Chen, J., Hull, J. and Poulos, Z. (2019). Deep Hedging of Derivatives Using Reinforcement Learning.

Ciaburro, G. (2018). *Keras Reinforcement Learning Projects*. 1. ed. Packt Publishing Limited.

Ekenel, H. K. and Stiefelhagen, R. (2006). Analysis of Local Appearance-Based Face Recognition: Effects of Feature Selection and Feature Normalization. *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, 34–34.

Fujimoto, S., Hoof, H. van and Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods.

– (2020). *Github: Addressing Function Approximation Error in Actor-Critic Methods*. URL: https://github.com/sfujim/TD3 (visited on 03/29/2020).

Hagan, P., Kumar, D., Lesniewski, A. and Woodward, D. (Jan. 2002). Managing Smile Risk. *Wilmott Magazine* 1, 84–108.

Halperin, I. (2017). QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds.

Hasselt, H. V. (2010). Double Q-learning. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel and A. Culotta, 2613–2621. URL: http://papers.nips.cc/paper/3964-double-q-learning.pdf.

Heston, L. S. (1993). A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *The Review of Financial Studies* 6.2, 327–343.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv: 1207.0580 [cs.NE].

Hodges, S. D. and Neuberger, A. (1989). Optimal Replication of Contingent Claims under Transaction Costs. *Review of Futures Markets* 8, 222–239.

Hull, J. (2012). *Options, futures, and other derivatives*. 8. ed., Pearson internat. ed. Upper Saddle River, NJ [u.a.]: Pearson Prentice Hall.

Hull, J. and White, A. (2017). Optimal Delta Hedging for Options. *Journal of Banking & Finance* 82.

Islam, R., Henderson, P., Gomrokchi, M. and Precup, D. (2017). Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control.

Jiang, Z., Xu, D. and Liang, J. (2017). *A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem*.

Kolm, P. and Ritter, G. (2019). Modern Perspectives on Reinforcement Learning in Finance. *The Journal of Machine Learning in Finance* 1.1.

Leland, H. E. (1985). Option Pricing and Replication with Transactions Costs. *The Journal of Finance* 40.5, 1283–1301.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv: 1509.02971 [cs.LG].

Maas, A. L., Hannun, A. Y. and Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models.

Mastinsek, M. (2012). Charm-Adjusted Delta and Delta Gamma Hedging. *The Journal of Derivatives* 19.3, 69–76.

Merton, R. C. (1973). Theory of Rational Option Pricing. *The Bell Journal of Economics and Management Science* 4.1, 141–183.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. A. (2013). Playing Atari with Deep Reinforcement Learning. *CoRR*. URL: http://arxiv.org/abs/1312.5602.

OpenAI (2018). *OpenAI Five*. URL: https://openai.com/blog/openai-five/ (visited on 04/13/2020).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 2825–2830.

Peng, X. B., Andrychowicz, M., Zaremba, W. and Abbeel, P. (2017). Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *CoRR* abs/1710.06537. arXiv: 1710.06537. URL: http://arxiv.org/abs/1710.06537.

Polydoros, A. S. and Nalpantidis, L. (2017). Survey of Model-Based Reinforcement Learning: Applications on Robotics. *Journal of Intelligent & Robotic Systems* 86.2, 153–173.

QuantLib (2020). *A free/open-source library for quantitative finance*. URL: https://www.quantlib.org/ (visited on 04/16/2020).

Renault, E. and Touzi, N. (1996). Option Hedging and Implied Volatilities in a Stochastic Volatility Model. *Mathematical Finance* 6.3, 279–302. DOI: 10.1111/j.1467-9965.1996.tb00117.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9965.1996.tb00117.x.

Ritter, G. and Kolm, P. (Jan. 2018). Dynamic Replication and Hedging: A Reinforcement Learning Approach. *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3281235.

Rubinstein, M. (1976). The Valuation of Uncertain Income Streams and the Pricing of Options. *The Bell Journal of Economics* 7.2, 407–425.

Ruf, J. and Wang, W. (2019). Neural networks for option pricing and hedging: a literature review.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I. and Abbeel, P. (2015). Trust Region Policy Optimization. arXiv: 1502.05477 [cs.LG].

Silver, D., Lever, G., Hees, N., Degris, T., Wierstra, D. and Riedmiller, M. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (7587), 484–489. URL: https://doi.org/10.1038/nature16961.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. and Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, I–387–I–395.

Strehl, A., Li, L., Wiewiora, E., Langford, J. and Littman, M. (Jan. 2006). PAC model-free reinforcement learning. *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning* 2006. DOI: 10.1145/1143844.1143955.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. A Bradford Book.

U.S. Department of the Treasury (2020). *Daily Treasury Yield Curve Rates*. URL: https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=yield (visited on 04/10/2020).

Watkins, C. J. C. H. (1989). Learning from Delayed Rewards. PhD thesis. King's College.

Whalley, A. E. and Wilmott, P. (1997). An Asymptotic Analysis of an Optimal Hedging Model for Option Pricing With Transaction Costs. *Mathematical Finance* 7.3, 307–324.

Xu, B., Wang, N., Chen, T. and Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. *ArXiv* abs/1505.00853.

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z. and Dasgupta, S. (2019). *Model-based Deep Reinforcement Learning for Dynamic Portfolio Optimization*.

Zakamouline, V. I. (2002). European Option Pricing and Hedging with both Fixed and Proportional Transaction Costs.

– (2006). Optimal Hedging of Options with Transaction Costs.